SDC

TM-1003 009 00

Milestone 11

OSAS-A Modified for Augmentation (SOSAS)

[illegible]

TM-1003/009/00

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract AF 19(628)-1648, Space Systems Division Program, for Space Systems Division, AFSC.

Milestone 11

OSAS-A Modified for Augmentation (SOSAS)

By

C. L. Hill

15 March 1963

Approved

J. B. Munson

SYSTEM DEVELOPMENT CORPORATION

2500 COLORADO AVE.

SANTA MONICA

CALIFORNIA

A-1159

IDENTIFICATION

A. Title: OSAS-A Modified for Augmentation (SOSAS) - Ident 05C, Mod AC

B. Programmed: 20 December 1962
Control Data Corporation
C. L. Hill, System Development Corporation

C. Documented: 15 March 1963

PURPOSE

SOSAS allows a programmer to write programs for the 160-A computer in symbolic notation with minimum regard for ultimate storage locations assigned a program. It also allows a programmer to write instructions using mnemonic symbols, which are easier to remember and to later interpret than are the octal machine code equivalents.

SOSAS is available in two versions with distinct Input/Output configurations:

A. Version A of SOSAS assembles symbolic programs from cards via the 088 on the 1610, provides a binary output on cards via the 533 on the 1610, and listable output appears on the 1612 printer.

B. Version B of SOSAS assembles symbolic programs from cards via the 167 card reader, provides a binary output on magnetic tape, and listable output appears on the 166 printer.

The primary difference between SOSAS and OSAS-A is the handling of a System Symbol Table by SOSAS.

USAGE

A. General Description

SOSAS is a two pass assembler. If the condensed representation of the symbolic program exceeds available core storage, it is dumped onto magnetic tape for intermediate storage.

This intermediate information is read back in for the assembler's second pass. During the first pass of the assembly process, the assembler reads in each line of symbolic information, scans the various fields, and stores a condensed representation of each line into a reserved block of storage. When that storage block is full, its contents are dumped onto magnetic tape. This is the intermediate output. Other steps which occur during the first pass include the assigning of values to location symbols, the entering of symbolic quantities into the symbol table, translating of operation codes, and advancing the current location counter. During the second pass the assembler analyzes and converts the intermediate information. It transmits each line of listable output and grouping of assembled binary words to the appropriate output device.

B. Special Halt and Suppress Functions

A programmer may halt the assembly process temporarily at any point by inserting a WAI pseudo-op at that point in his program or setting a stop, making whatever changes he wishes, and continuing. He may also suppress either the listable or binary output by inserting a SUPA or SUPB pseudo-op in his program.

C. Program Relocation

All programs assembled by SOSAS are relocatable. Relocation is accomplished by specifying a relocation constant at load time. This relocation constant is added, under control of the loader, to the storage address assigned to relocatable words in a program assembled by SOSAS. Relocatable words are words that may be stored in any location in memory. Each word that is capable of being relocated must be assembled under control of the ORG-PRG counter. Therefore, any word or group of words that the programmer wishes to be relocatable must be associated with the ORG or PRG pseudo-op as shown in the example in Appendix B. In addition, words consisting entirely of addresses that refer to relocatable words in the symbolic program must be modified by the relocation constant. Words that are stored in low core locations ($0000\text{-}0077_8$) in any bank are normally non-relocatable and are assembled under control of the CON counter. Storage locations assigned to these words are not incremented by the relocation constant.

Words that will not be modified by the relocation constant are as follows:

1. Words that contain an op code; or op code, address and additive fields.

2. Words that consist of entirely numeric address and additive fields.

3. Words that refer to addresses of words assembled under control of the CON counter.

D. Symbol Table Manipulation

SOSAS has the capability to accept a symbol table and include it as a semi-permanent part in an "updated" version of SOSAS, allowing for further updating in the form of additional new symbols. When SOSAS is updated with a symbol table, both a listing and a binary version of the updated symbol table are produced.

A symbol table in the form of symbolic cards will be accepted by SOSAS upon the setting of the SLJ-2 switch. The symbol (not greater than six characters) will be punched, left justified in col. 2-7. The op. code, EQU, will be punched in col. 10-12. A four digit octal number must be punched in col. 15-18. This is the location assigned to the symbol.

A deck of symbol table cards may be preceded by an IDNT card, which will be used in the list heading. The IDNT card must have an op. code, IDNT, punched in col. 10-13. Any four character identification may be punched in col. 15-18. If an IDNT card is not present, SOSAS will identify the symbol table listing with the information on the IDNT card of the last updating which included an IDNT card, or if no IDNT card has ever been used, no special identification will be printed.

The deck of symbol table cards must be terminated by an END card. This card consists of the op. code, END, punched in col. 10-12.

The symbol table listing consists of a header, SYMBOL TABLE XXXX (XXXX is omitted if no IDNT card is used), and a list of the symbols with either octal equivalences.

The binary version of the symbol table is punched on cards by Version A and written on magnetic tape unit 4 in binary card format by Version B. The cards are of the same format as described on page 13. No relocation bits are present. The starting address of the first card is 4000_8. Each symbol is represented by four words on the card, three words expressing the BCD characters of the symbol and one word indicating the binary equivalence of the symbol. The last word is always 7776_8.

E. Symbolic Input

A symbolic program to be assembled by SOSAS is read in from punched cards via the IBM088 card reader on the CDC1610 Control Unit by Version A and via the CDC167 card reader by Version B.

All input is BCD in the computer. Remarks and other pseudo-ops may be located anywhere in the program; however, each program must end with a WAI or an END pseudo-op. The location, address, and additive fields are preceded by a plus, a minus, or a blank. (A blank indicates a positive field.) The description of each field in a line of symbolic input is given below.

1. Location Field

A location symbol may be a maximum of six alpha-numeric characters. The sign (or blank) is not a part of the location field. Numeric characters may be either octal or decimal. Resultant octal values equal to or greater than $10{,}000_8$ may be used as location symbols. Values less than $10{,}000_8$ will be entered in the symbol table but can not be referenced by other instructions. Four-digit decimal numbers that result in five-digit octal values are valid symbols.

2. Operation Code Field

This field may contain any of the symbolic op codes or paeudo-ops listed in Appendix A.

Two-digit octal machine instruction codes may also be used if they are left-justified within the field.

3. Address Field

This field may contain an octal number, a decimal number followed by a D, or a location symbol. Any location symbol used in an address field must either appear in the location field somewhere in the program or have been previously defined in a symbol table used in "updating" SOSAS.

4. Additive Field

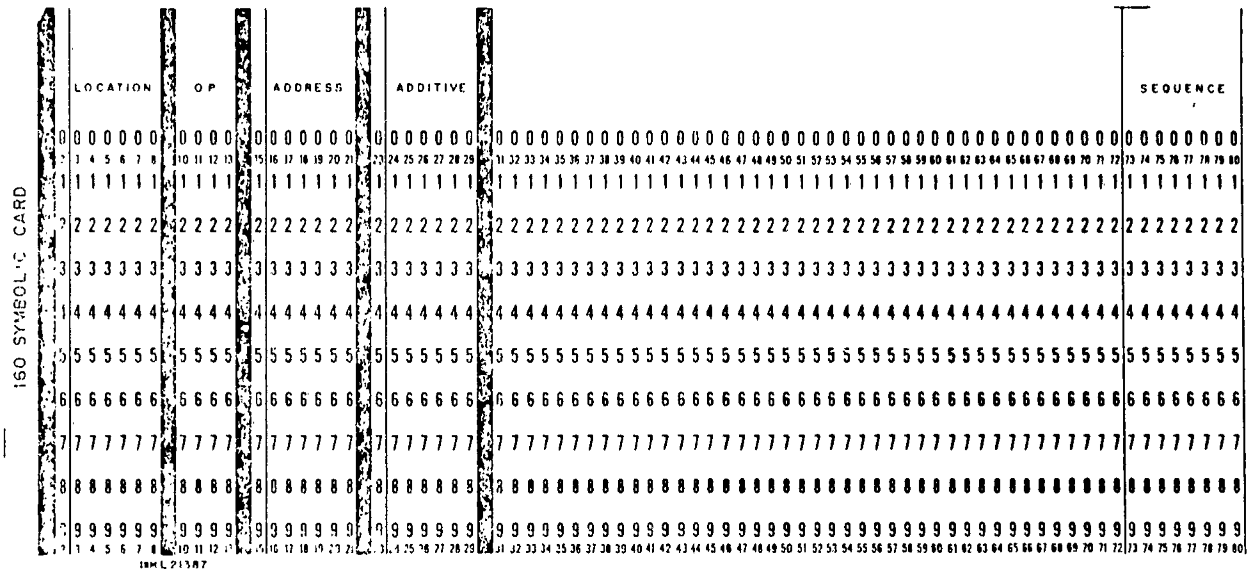
This field is used to increment the address specified in the address field. This field may contain a location symbol, an octal integer, or a decimal number followed by a D. Any location symbol used in an additive field must either appear in the location field somewhere in the program or have been previously defined in a symbol table used in "updating" SOSAS.

Information specified by the additive field will be added algebraically to information specified in the address field. Location symbols appearing in either field are represented by their assigned machine addresses during address computation. The resultant sum is added or subtracted from the current address to obtain a new address somewhere in the program. The resultant sum can never exceed $\pm 77_8$ except for Memory Address Mode instructions, the Return Jump Instruction, or the Selective Jump Instructions. If a minus sign is punched in the sign position preceding the location field of a line, the current address is ignored in processing the address and additive fields for that line.

5. Comments Field

This field may contain up to 50_{10} characters.

F. Card Symbolic Format



Columns	Contents
2	Minus, plus, blank or first character of Location Symbol
2-8	Location symbol
10-13	Op Code
15	Minus, plus, blank or first character of Address Field
16-21	Address Field
23	Minus, plus, blank or first character of Addition Field
24-29	Additive Field
31-80	Comments Field

Contents of the location, address, and additive fields may be left-justified, i.e., may start in columns 2, 15 or 23 respectively.

G. Arrangements of Symbolic Deck for Assembly

No special start or identification card is required. A remarks card may be included to identify the program but is not necessary. Blank cards are ignored; they do not affect the object program. The last card must be an END or a WAI card. Decks may be followed by 3 blank cards to assure reading of the last card.

Decks may be stacked for assembly by placing several programs in the reader at once, one program following another with no special cards between.

H. Mnemonic Codes for SOSAS

SOSAS recognizes all of the normal mnemonic operation codes for the 160-A computer. In addition, it recognizes certain pseudo-ops and 15 special relative codes. The special relative codes are relative instructions which do not specify direction. An R is substituted for the terminal B or F in the symbolic program and SOSAS determines the proper direction automatically as explained on page 10. For example, LPR may be coded instead of LPF or LPB. The special relative mnemonics are: ADR, AOR, LCR, LDR, LPR, LSR, NJR, NZR, PJR, RAR, SBR, SCR, SRR, STR, ZJR.

Appendix A contains a complete list of all mnemonic codes recognized by SOSAS, including normal operation codes, pseudo-ops, and the special relative codes, arranged in alphabetic order. To provide compatibility with previous 160 assembly programs, SOSAS accepts the old Shift A and Logical Sum instructions.

I. SOSAS Pseudo- Instructions

The SOSAS assembly program recognizes 16 pseudo-instructions (pseudo-ops). Pseudo-ops are not interpreted as machine language instructions; they provide the programmer with a means of controlling the assembly of a symbolic program, and are recognized only by the assembly program. Three of these, the ORG, PRG, and CON pseudo-ops, are included in the pseudo-op repertoire because of the relative addressing feature of the 160-A computer. These three pseudo-ops determine the storage location of each assembled line in the object program by specifying the contents of the two assembly program location counters - the ORG-PRG counter and the CON counter. Addresses assigned to each line during assembly are under control of whichever counter is active at that time. The active location counter is incremented after the current line is assembled.

The CON counter is actuated by a CON pseudo-op and the ORG-PRG counter is actuated by either an ORG or a PRG pseudo-op. At the beginning of a program, SOSAS automatically sets the ORG-PRG counter to 100_8 and the CON counter to 0. Relocatable lines are assembled under control of the ORG-PRG counter; non-relocatable lines (those that are to be placed in low core) are assembled under control of the CON counter.

1. ORG-PRG Counter

   This counter is activated by either an ORG or a PRG pseudo-op. The counter is set to the value of the algebraic sum of the address field and the additive field EXCEPT when both these fields are blank. The instruction immediately following an ORG or a PRG pseudo-op is assembled to the location contained in the ORG-PRG counter as a result of the pseudo-op. If neither of the pseudo-ops appear at the beginning of a program, theORG-PRG counter is set to 0100_8. In case the additive and address (AA) fields are blank, the following occurs:

   a. ORG causes the ORG-PRG counter to be set to 0.

   b. PRG causes the ORG-PRG counter to resume control and to continue counting from the previous value contained in the counter.

2. CON Counter

   This counter is initially set to 0 if no CON pseudo-op occurs at the beginning of a program. When a CON pseudo-op is encountered, the CON counter is set to the algebraic sum of the address and additive (AA) fields. This sum must never be greater than 77_8. As long as the CON counter remains active, it is incremented by one except when it is reset by the AA fields of a CON pseudo-op. If the AA fields of a CON pseudo-op are blank, counting begins from the previous value. The legal range of a CON counter is from 0 through 77_8. Locations higher than 77_8 that are specified by a CON pseudo-op are flagged as range errors.

3. Symbol Table

   Each symbol that appears in the location field of a line in a symbolic program is assigned a numeric value by SOSAS and is placed in a symbol table. This symbol table consists of two parts; a variable portion and a constant portion.

   The variable symbol table contains the numeric value of all symbols that refer to words that are relocatable. The constant symbol table contains the numeric value of all symbols that are non-relocatable. Symbols that are assigned to constants or low core addresses and System Symbols are placed in the constant symbol table. The capacity of the entire symbol table is 1000_{10}. The following formula may be used for computing the allowable variable or constant symbol table capacity in a program to be assembled.

$$S_v \quad + \quad 2S_c \leq 1000_{10}$$

Where S_v = total number of variable symbols
and S_c= total number of constant symbols

4. Pseudo-Ops

a. ORG - Causes the ORG-PRG counter to assume control. This pseudo-op causes the ORG-PRG location counter to assume the value of the algebraic sum of the address and additive (AA) fields. This value is assigned as the location to which the next instruction (after an ORG) will be assembled. Each word assembled under this pseudo-op will be flagged as relocatable on the binary output card or card image. Symbols appearing in either AA field must be defined before ORG is executed. An ORG should not be used to continue an assembly. Blank AA fields set the ORG-PRG counter to 0. See Appendix B for example.

b. PRG - has all the properties of ORG except that it is used principally to continue an assembly. A PRG with blank AA fields reactivates the ORG-PRG counter. If a PRG with blank AA fields is the first instruction of a program, the counter starts at 100_8. See Appendix B for example.

c. CON - Controls the CON counter as the PRG controls the ORG-PRG counter. Symbolic locations defined under control of the CON pseudo-op may be referenced with no-address, direct, or indirect address mode instructions only. Avoid forward, backward, and relative mode instructions when referencing such symbolic locations. Words assembled under control of the CON pseudo-op are non-relocatable, and may be stored only in low core. See Appendix B for example.

d. BLR, BSS - will advance the location counter currently in control by the amount specified in the address plus additive field. If a symbol is given in the location field, that symbol will be assigned to the first numberic address in the block. Care must be taken to ensure that CON counter range is not exceeded.

e. WAI - Will cause the assembly program to stop and allow for insertion of a new tape for input. The END pseudo-op may be simulated here by entering a non-zero quantity into the A-register and running.

f. END - Will cause the assembly program to prepare for the second pass. During binary output, a transfer card is produced. An END pseudo-op with blank AA fields will cause the loader to transfer program control to address 0000. Otherwise, an END pseudo-op will transfer program control to the address designated by the algebraic sum of the AA fields.

NOTE: An END or WAI pseudo-op must terminate the program being assembled. If a WAI is the last instruction, the END pseudo-op must be generated manually in order to complete the assembly process.

g. EQU - Assigns the algebraic sum of the address and additive fields of the EQU pseudo-op to the symbol given in the location field, and places this symbol and its numeric value in either the variable or the constant symbol table. The address and additive fields may be either numeric or symbolic. If both fields are numeric, then the symbol in the location field is assigned to the constant symbol table and is not relocatable. (To maintain program relocatability, numeric locations should refer only to low core.) If both address and additive fields are symbolic, the algebraic sum of the two fields are equated to the symbol in the location field. If either address or additive symbol is relocatable, then the symbol in the location field will be relocatable. If both symbols are non-relocatable, then the location field symbol will be non-relocatable. (Constants or low-core addresses). If the address and additive fields consist of a symbol and a numeric value, in either order, then the relocatability of the location field symbol is determined by the relocatability of the symbol in either field.

h. REM - Takes all that follows the additive field as remarks and is ignored by the assembly program. A REM instruction will not cause the location counter to advance. A maximum of 50_{10} characters is permitted in the remarks field.

i. BNKX - Will generate a binary bank card that will cause the instructions following the pseudo-op to be loaded into bank X. (X must be an octal digit). This pseudo-op does not affect the assembly of a symbolic program; it is a signal to the loader.

j. SUPA - Suppresses the assembly listable output.

k. SUPB - Suppresses the binary output.

l. BCD - Causes a contiguous string of characters to be assembled 2 BCD characters per word.

m. BCDR - Causes a contiguous string of characters to be assembled 1 BCD character per word.

n. FLX - Causes a contiguous string of characters to be assembled 2 flex characters per word.

o. FLXR - Causes a contiguous string of characters to be assembled 1 flex character per word.

The following rules affect BCD, BCDR, FLX, and FLXR pseudo-ops. A maximum of 50_{10} characters is permitted by the BCD, BCDR, FLX, and FLXR pseudo instructions. The characters must be in the remarks field; a character count must appear in the address field.

J. Assembly Rules for Input

1. A line containing information in the remarks field only will be ignored during assembly but will appear in proper order with the listable output.

2. If the OP field of a line is blank, then the address and additive fields are added algebraically and stored as one 12-bit number. (If the result is negative, the line will assemble without error, but the resultant machine word will be incorrectly modified by any relocation constants).

3. If the OP field is non-relative, the address and additive fields are added algebraically to form a 6-bit low-core machine address or a 6-bit constant. If the sum of the address and additive field exceeds 77_8 the line is flagged as a range error in the listable output and the low-order 6-bits of the instruction are set to zero.

4. If the OP field is relative (last character is F, B, or R) the contents of the location counter is subtracted from the algebraic sum of the address and additive fields. (A minus sign punched in the sign position of the location field indicates that a line is to be assembled as in rule 2). For F-type op codes, if the result of this operation is positive, the result is directly inserted in the

low order 6-bits of the resultant 12-bit machine instruction. If the result is negative, the low order 6-bits are cleared to zero and the line is flagged as a possible range error in the listable output. For B-type op codes, if the result of the subtraction is negative, the result is complemented before being combined with the high order 6-bit machine op code. If the result is positive, the low order 6-bits are cleared and the line flagged as a possible range error in the same manner as a negative result for F-type op codes. The possibility of range errors may be reduced by substituting for F- or B-type op codes an R-type op code, which is recognized by SOSAS. If, for example, a programmer wishes to use one or the other of the two relative machine codes (F or B), but does not know at the time the program is written whether the desired reference is forward or backward, an R-type code may be used. An R-type (special relative) op code forces SOSAS to examine the result of the subtraction of the contents of the location counter from the sum of the address and additive fields to determine the correct relative machine op code. If the result is positive, the resulting machine op code will be an F-type; if the result is negative, the resulting machine op code will be a B-type. (JPR and ERR are not considered R-type op codes and will not be recognized as such by SOSAS). For all three types of relative op codes, if the difference between the current location and the sum of the address and additive fields is greater than $\pm 77_8$, the low order 6-bits of the resulting machine instruction are cleared to zero and the line is listed with a range error flag.

X. Listable Output

1. Listable output is on the 1612 by Version A and on the 166 by Version B.

Columns	Contents
2 - 5	Error flags
7 - 10	Octal location (location in core to which the line of data was assembled)
13 - 16	Octal contents (contents of octal location)
19	Blank or minus sign
20 - 25	Location symbol
27 - 30	Op code
32	Plus sign, minus sign or blank
33 - 38	Address
40	Plus sign, minus sign or blank
41 - 46	Additive
48 - 97	Comments

2. Error Flags
SOSAS has eight error codes which may appear on an assembly listing.

Error Code	Explanation
C	CON location out of range (beyond first 100_8 locations)
E	E term of assembled machine op code greater than 77_8
L	Undefined symbol in location field
M	Symbol defined more than once
O	Illegal op code
U	Undefined symbol in address field
V	Undefined symbol in additive field
X	Illegal character

L. Binary Output

Binary output is punched on cards by the 533 on the 1610 by Version A and it is written on magnetic tape unit 4 in card format for off-line punching by Version B. All binary output produced by SOSAS is relocatable. It will include program cards, bank cards, and a transfer card for each program.

Format for binary program cards, bank cards, and transfer cards:

1. Program Card

Columns	Rows	Information
1	7, 9, 12	Binary program card indicators
1	8	If check sum is to be ignored
1	0-6	Word count
2		Starting Address
3		Check sum of other 79 columns
4-9		Designator bits for words in columns 10 through 80 that must be modified by relocation constant. A punch indicates that word will be modified at load time if relocation constant is specified. A punch in row 9, column 9, indicates that words in column 10 through 80 will not be relocated. (A 12-punch in column 4 indicates that word 10 is modified; an 11-punch in column 4 indicates that word 11 is modified; a 0-punch in column 4 indicates word 12 is modified. Ordering of word-bit correspondence proceeds in sequence down each columns, 4 through 9.)
10-80		Machine instructions, addresses or constants.

2. Bank Card	Rows	Information
1	7, 9, 11	Bank card indicator
1	8	If checksum is to be ignored
2	7-9	Bank number
3		Checksum
3. Transfer Card		
1	7, 9, 12	Binary transfer card indicator
1	8	If checksum is to be ignored.
2		Transfer address
3		Checksum

M. Operating Instructions for Assembly Process

1. Magnetic Tape

   a. Unit #2 - Intermediate input/output

   b. Unit #4 = Binary output (Version B only)

2. All SOSAS bi-octal paper tapes should checksum to zero when loaded. The last word on every tape is the complement of the checksum of the rest of the tape, thus giving a zero checksum when loaded.

3. To Assemble without Updating Symbol Table

   a. Load SOSAS bi-octal paper tape at 0. Checksum should be 0.

   b. Master clear.

   c. RUN from 0.

   d. At stop 2062_8 program has been assembled correctly. RUN to rewind unload tape units or master clear and RUN to assemble next program. If stop 2061_8 occurs, certain lines of symbolic input were in error. A-register displays number of lines in error. RUN to stop at 2062_8.

4. <u>To Update Symbol Table - SLJ-2 Set</u>

   a. Load SOSAS bi-octal paper tape at 0. Checksum should be 0.

   b. Master clear.

   c. Ready symbol table cards in card reader.

   d. RUN from 0 (making sure SLJ-2 is set).

   e. After halt at 3744_8:

      (1) If it is desired to assemble with new updated symbol table without punching a new version of SOSAS, ready symbolic cards in card reader, ready printer, ready punch for Version A, reset SLJ-2 and run. It should be noted, that, when using Version B, binary output will be written on the same tape that the binary symbol table was written on. No tape manipulation is necessary. The assembly process will write the binary output on the tape without positioning.

      (2) If a new version of SOSAS, including the symbol table just read in, is desired, turn paper tape punch motor on and run. A bi-octal paper tape of the modified SOSAS is punched, an updated list of the symbol table is punched on binary cards. A second halt now occurs at location 4366_8.

         (a) If no assembly is being run, processing is complete.

         (b) If an assembly is now desired, ready symbolic cards in card reader, ready printer, ready punch for Version A, and run. For Version B, no tape manipulation is necessary.

5. <u>To halt assembly program prior to completion</u>

   a. Set stop switch 1. Do not take out of RUN position to stop assembly program.

   b. To resume assembly, restore stop switch 1 and RUN.

   c. To restart assembly program, master clear and RUN.

6. <u>Stops</u>

   a. Normal Stops:

(1) 0776_8 will occur if Jump Switch 1 has been set. Occurs after the first pass.

(2) 2062_8 Final stop. The assembly has been completed correctly. Master clear and RUN to assemble next program.

(3) 2207_8 The WAI pseudo-op has been encountered. Position next symbolic input portion and RUN to continue assembly. To simulate an END pseudo-op, enter some non-zero quantity into the A-register and RUN.

(4) 3744_8 Stop after reading in Symbol Table cards (SLJ-2 set). To punch updated version of SOSAS, punch on binary cards and list the updated Symbol Table, RUN. To assemble using updated Symbol Table but not punch updated SOSAS or print and punch Symbol Table, reset SLJ-2 and RUN.

(5) 4366_8 Updating process is completed. To assemble using updated SOSAS, ready input/output equipment and RUN.

b. Assembly Error Stops:

(1) 2061_8 Line error display. The number of program lines containing errors is displayed in the A-register. RUN to proceed to final stop 2062_8.

(2) 0364_8 or 3666_8 Symbol table is too large for computer to handle. Either the number of symbols must be decreased or the program must be assembled in parts; symbols common to more than one part must be EQU'd to the address immediately after the last address of the preceding portion. Maximum symbol table capacity is 1000_{10}.

(3) 3734_8 Illegal card in Symbol Table input. To recover in Version A, run cards out of 088; first card out is the bad card. To recover in Version B, top card in stacker is bad card. For both versions, correct error; make remaining cards ready in reader, beginning with corrected card; start at 3711_8.

c. Version A Input/Output Error Stops:

(1) 5042_8 533 punch not ready. RUN after correcting error.

(2) 5067_8 088 reader not ready. RUN after correcting error.

(3) 5241_8 Write parity error on intermediate I/O on magnetic tape. RUN to rewrite. Zero A-register and RUN to disable parity for remainder of record; then continue normal processing.

(4) 5310_8 Read parity error on intermediate I/O on magnetic tape. RUN to reread. Zero A-register and RUN to disable parity for remainder of record; then continue normal processing.

d. Version B Input/Output Error Stops:

(1) 5010_8 Write parity error on binary output on magnetic tape. RUN to rewrite. Zero A-register and RUN to disable parity for this record; then continue normal processing.

(2) 5041_8 167 card reader not in operating condition. A-register displays 10XX where XX indicates octal status of 167 as given below. Note contents of P-register. Correct 167 error condition and master clear. Add one to noted contents of P, enter sum in P-register, zero A-register and RUN.

167 Status Response Codes

(a) 01 Hopper empty

(b) 02 Stacker full

(c) 03 Feed failure

(d) 10 Program error

(e) 20 Amplifier failure

(f) 40 Motor power off

(3) 5161_8 Illegal code detected by 167 card read routine. A-register contains octal number of card column in error. Correct card and RUN.

(4) 5316_8 Write parity error on intermediate I/O on magnetic tape. RUN to rewrite. Zero A-register and RUN to disable parity for remainder of record; then continue normal processing.

(5) 5365_8 Read parity error on intermediate I/O on magnetic tape. RUN to reread. Zero A-register and RUN to disable parity for remainder of record; then continue normal processing.

RESTRICTIONS

A. Maximum number of System Symbols is 250_{10}.

B. Maximum number of all symbols allowable is 1000_{10}.

C. In Version B, the binary output tape (Unit #4) is not rewound before assembly. If this is required it must be done manually.

STORAGE REQUIREMENTS

SOSAS requires two full banks (Banks 0 and 1) to perform its operations. The entirety of Bank 1 is used to hold the symbol table. Bank 0 is used in the following manner:

A. 0000 - 5361_8 Program

B. 5362_8 - 7515_8 System Symbol Table and Intermediate Information Buffer. It should be noted that both the System Symbol Table and the Intermediate Information Buffer are variable in length. The maximum size of the System Symbol Table is 250_{10} which requires 1750_8 cells. The Intermediate Information Buffer always includes all of this area not required for the System Symbol Table.

C. 7516_8 - 7776_8 Input/Output Buffers.

VALIDATION TESTS

Because SOSAS is a modified version of OSAS-A, which is a much used program with no known outstanding problems, and because the only differences between SOSAS and OSAS-A are those which involve the System Symbol Table; validation tests on SOSAS were chiefly concerned with the handling and usage of the System Symbol Table.

The following tests were made:

A. SOSAS was updated with a System Symbol Table. This operation produced;

1. An updated SOSAS paper tape.

2. A System Symbol Table listing.

3. Binary cards of the System Symbol Table.

B. A short assembly was run with the updated operation. The program assembled contained symbols that were in the System Symbol Table.

C. The same assembly mentioned above was also run with the updated SOSAS paper tape produced in A.

D. A similar "second generation" test was performed. Steps A, B and C were performed starting with the updated SOSAS paper tape produced in A.

In Appendix "C" is the System Symbol Table listing produced in step A and a copy of the assembly listing produced in steps B and C. Since the two assembly listings are identical, only one reproduction is given.

REFERENCES

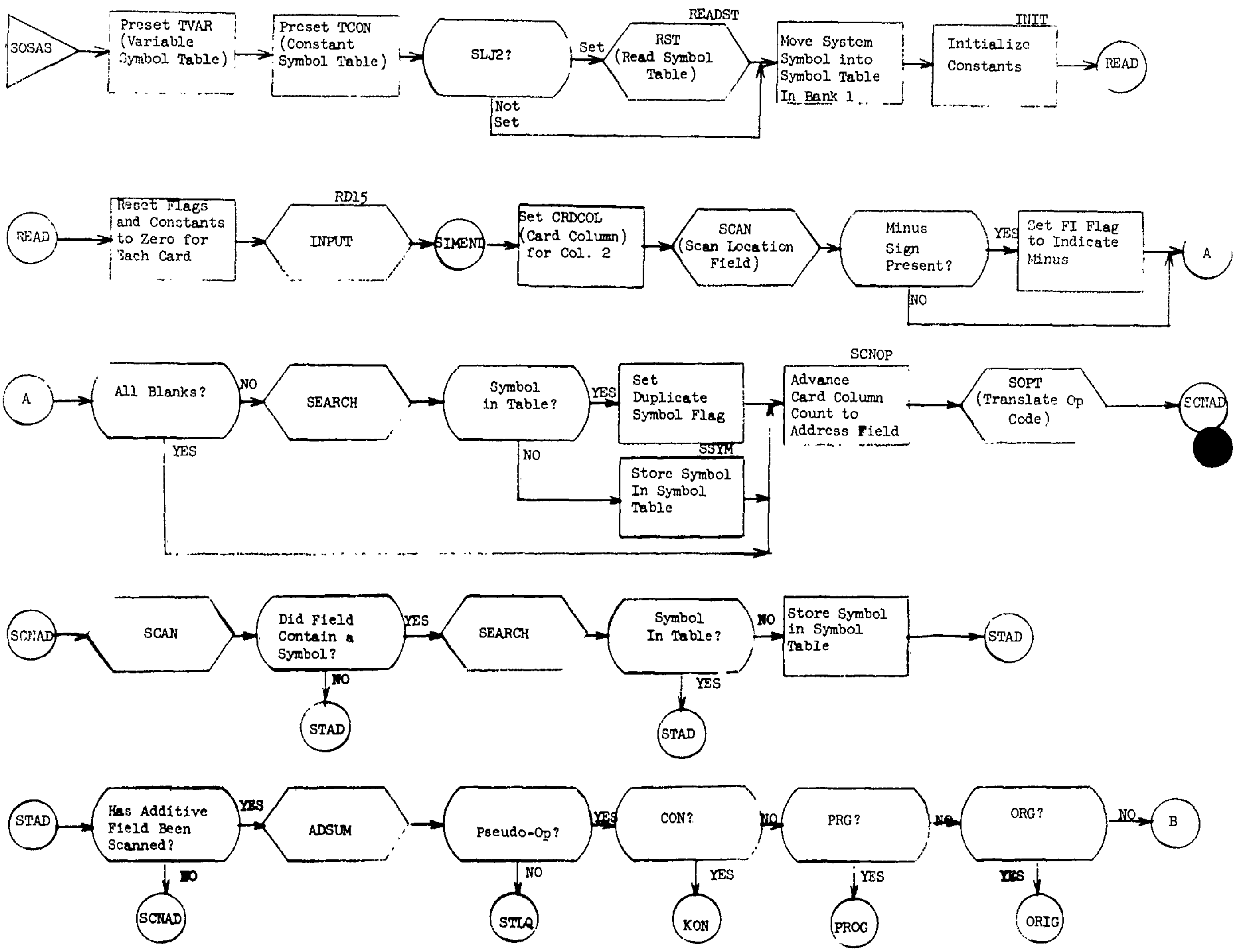
A. "OSAS-A, The 160-A Assembly System", Control Data Corporation, Pub. No. 507, May 1962.

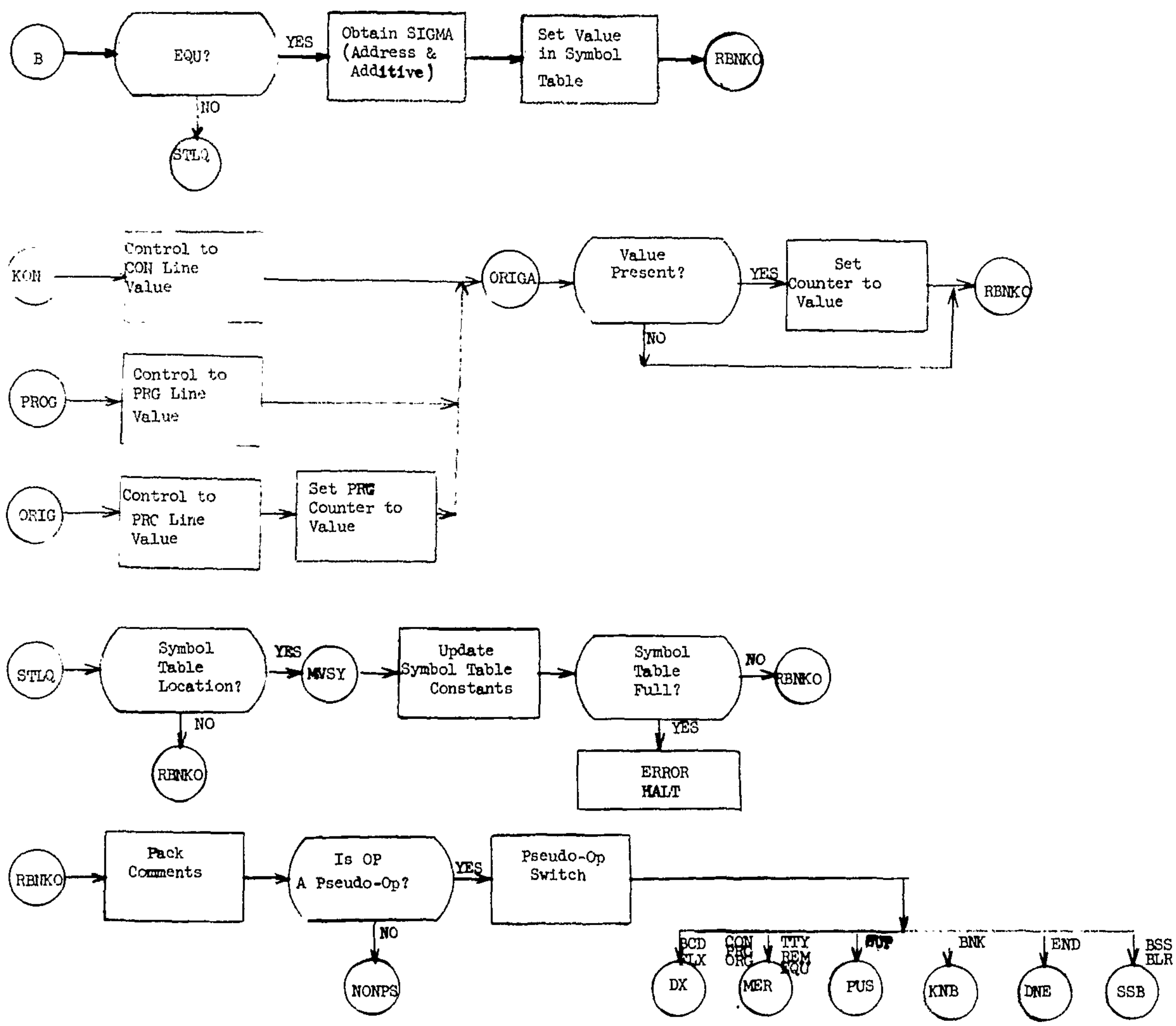
B. "Combined Milestone 3 and 4 for the Bird Buffer Utility Support System", System Development Corporation, TM-(L)-824/000/00, 5 November 1962.

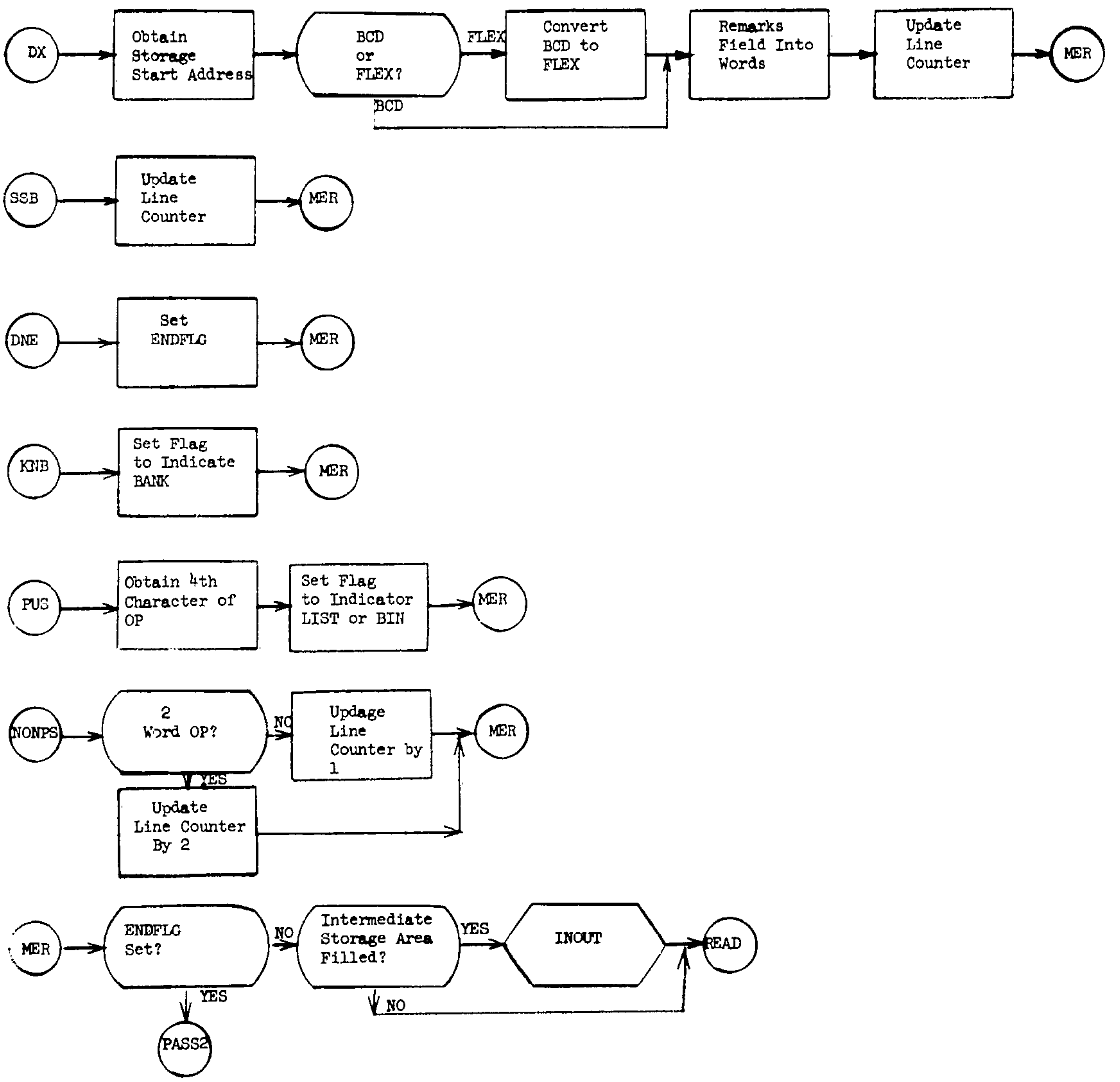
C. "General Description and Operating Procedures for SOSAS", System Development Corporation, N-(L)-19081/025/00*, 7 February 1963.

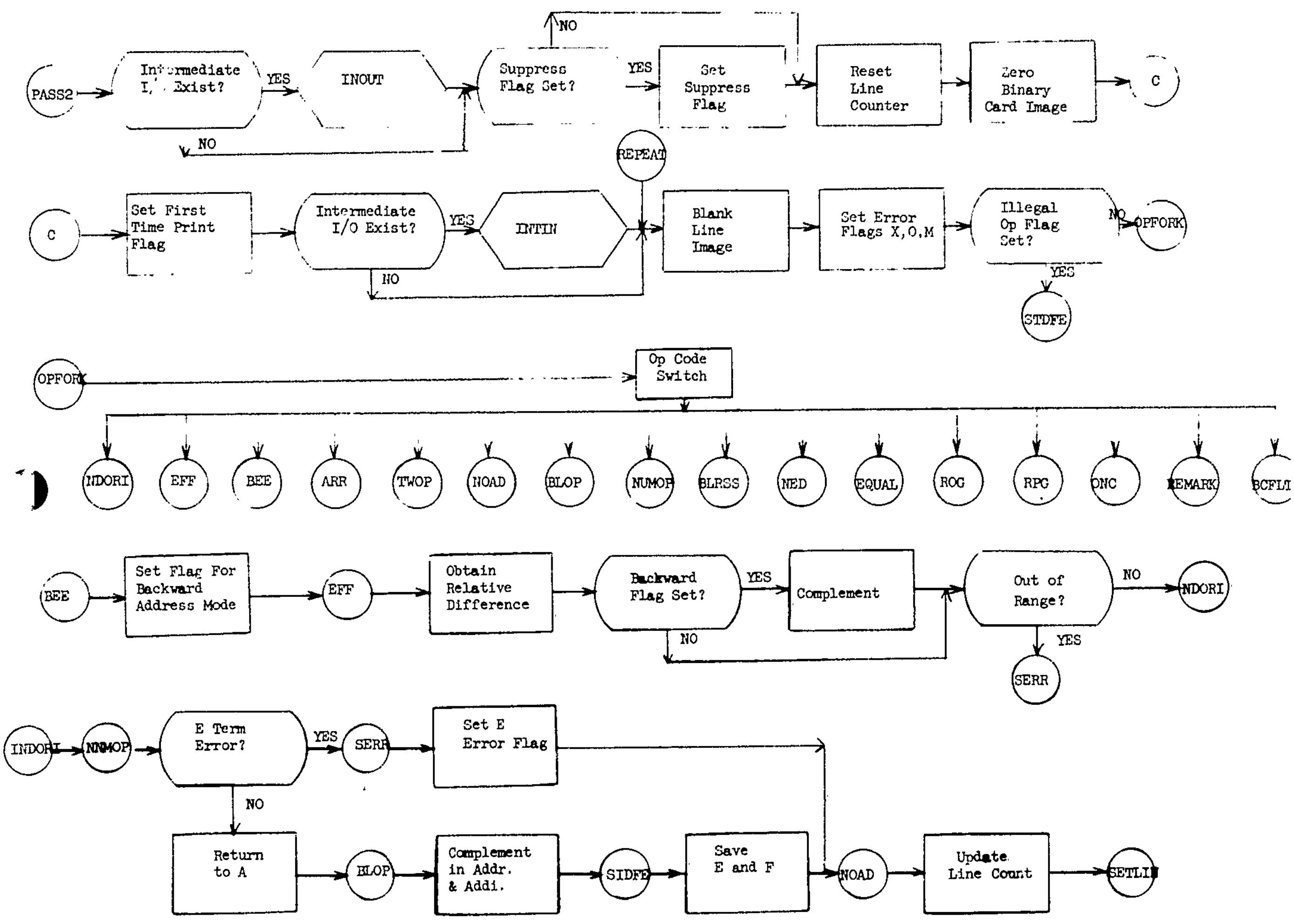
*N (Notes) are internal SDC documents and are ordinarily not released to outside companies.

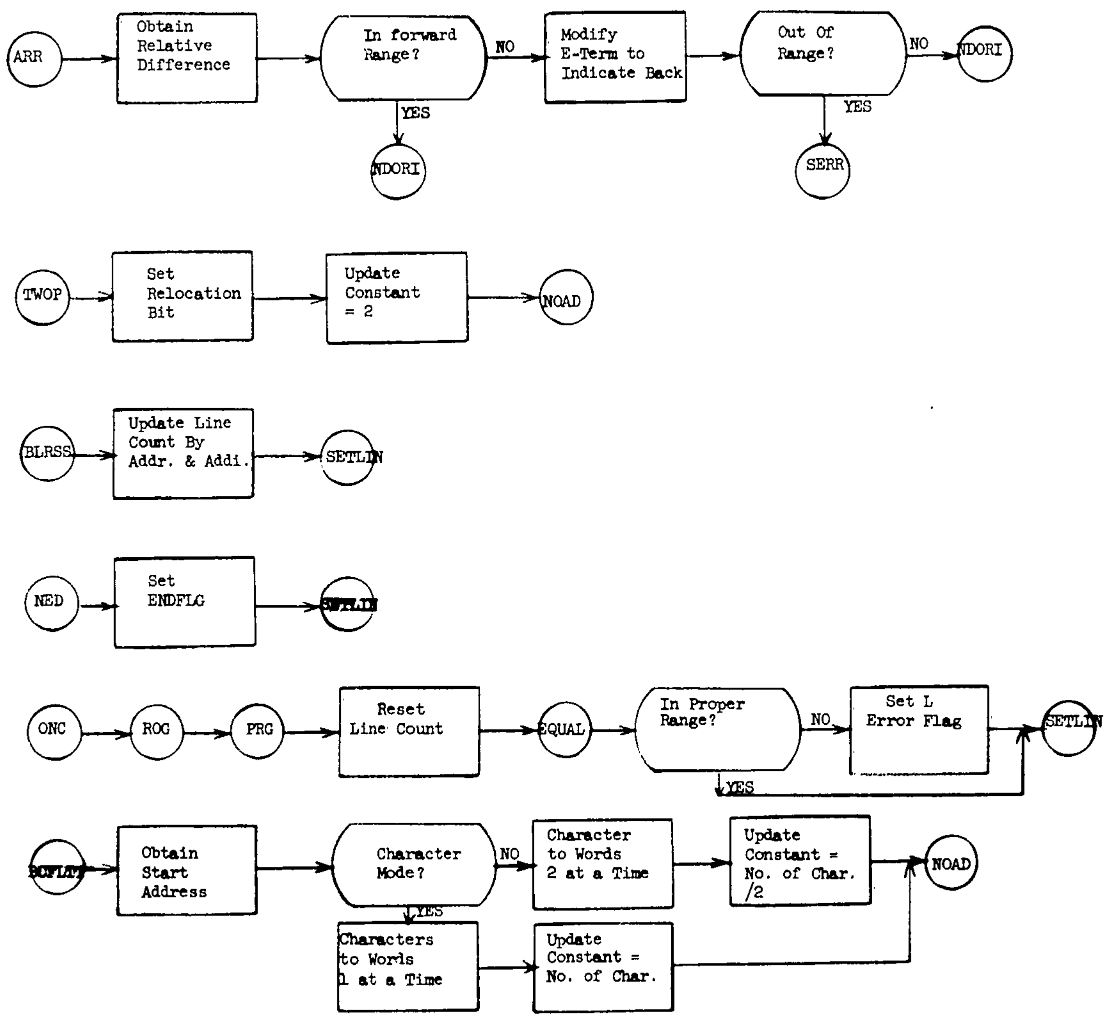
FLOW DIAGRAMS

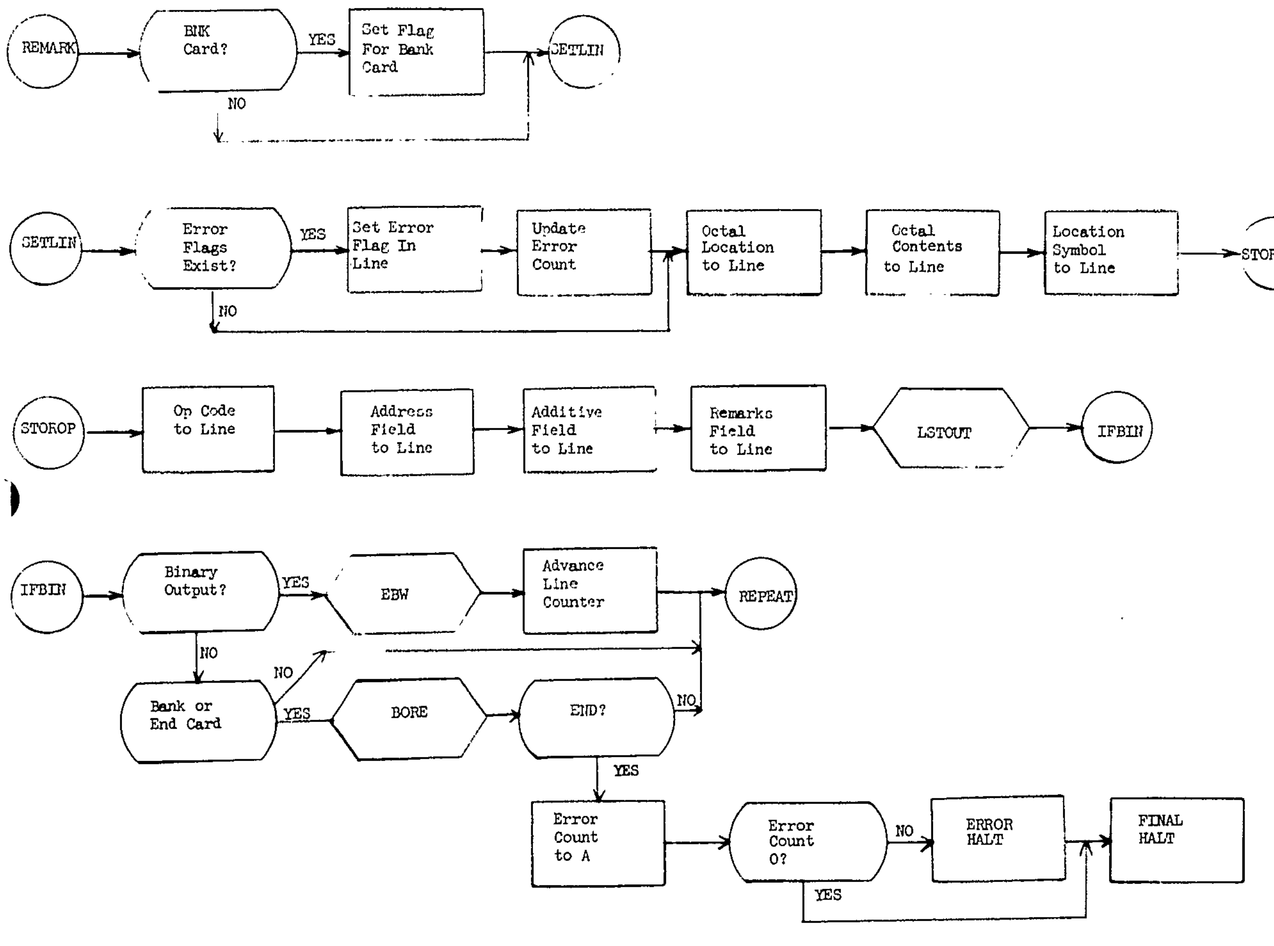


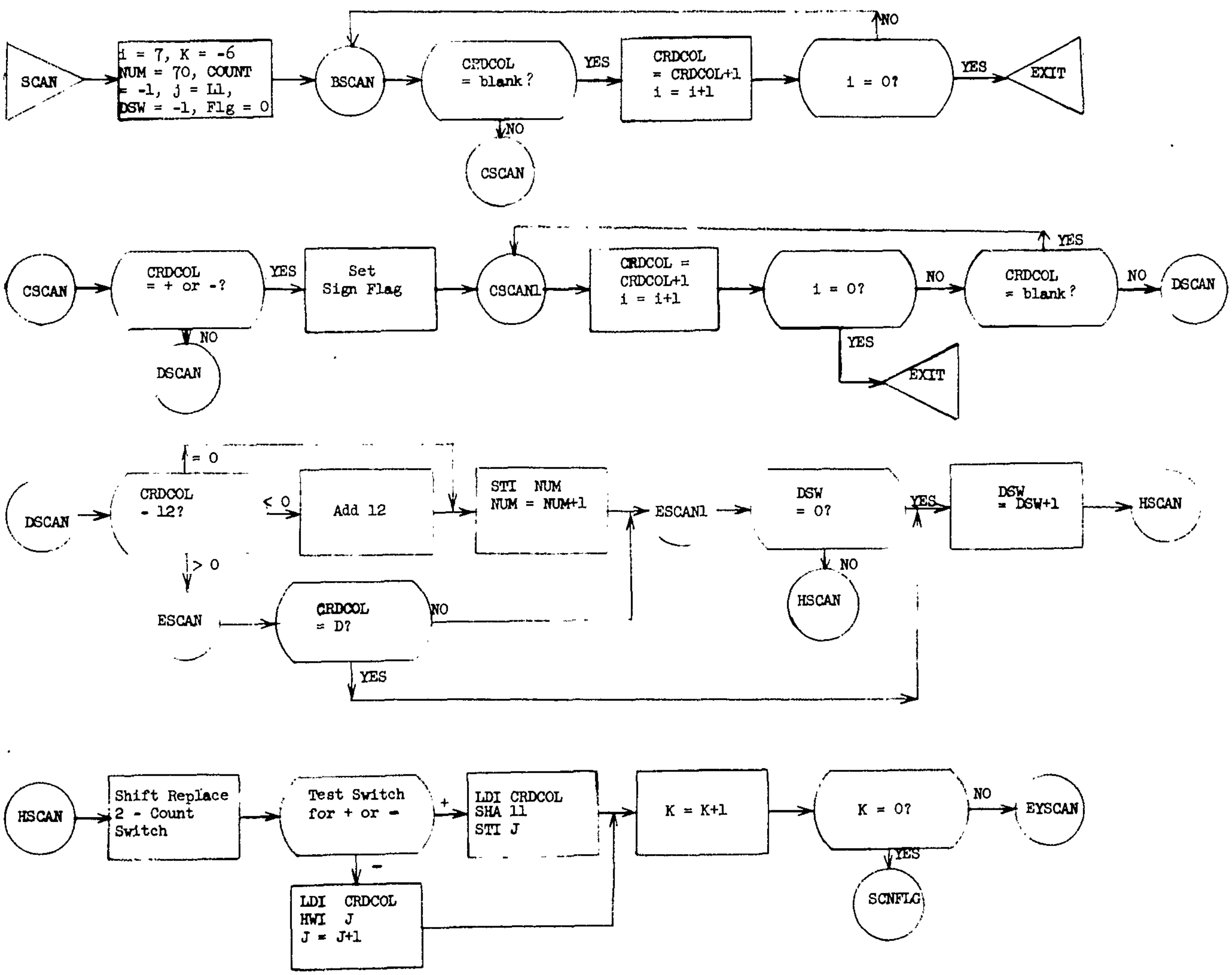

B
EQU?
YES
NO
Obtain SIGMA (Address & Additive)
Set Value in Symbol Table
RBNKO
STLQ
KON
Control to CON Line Value
PROG
Control to PRG Line Value
ORIG
Control to PRG Line Value
Set PRG Counter to Value
ORIGA
Value Present?
YES
NO
Set Counter to Value
RBNKO
STLQ
Symbol Table Location?
YES
NO
MVSY
Update Symbol Table Constants
Symbol Table Full?
NO
YES
RBNKO
RBNKO
ERROR HALT
RBNKO
Pack Comments
Is OP A Pseudo-Op?
YES
NO
Pseudo-Op Switch
NONPS
BCD
FLX
DX
CON
PRG
ORG
TTY
REM
EQU
MER
DUP
PUS
BNK
KNB
END
DNE
BSS
BLR
SSB


DX
Obtain Storage Start Address
BCD or FLEX?
FLEX
BCD
Convert BCD to FLEX
Remarks Field Into Words
Update Line Counter
MER
SSB
Update Line Counter
MER
DNE
Set ENDFLG
MER
KNB
Set Flag to Indicate BANK
MER
PUS
Obtain 4th Character of OP
Set Flag to Indicator LIST or BIN
MER
NONPS
2 Word OP?
NO
Updage Line Counter by 1
MER
YES
Update Line Counter By 2
MER
ENDFLG Set?
NO
Intermediate Storage Area Filled?
YES
INOUT
READ
NO
YES
PASS2

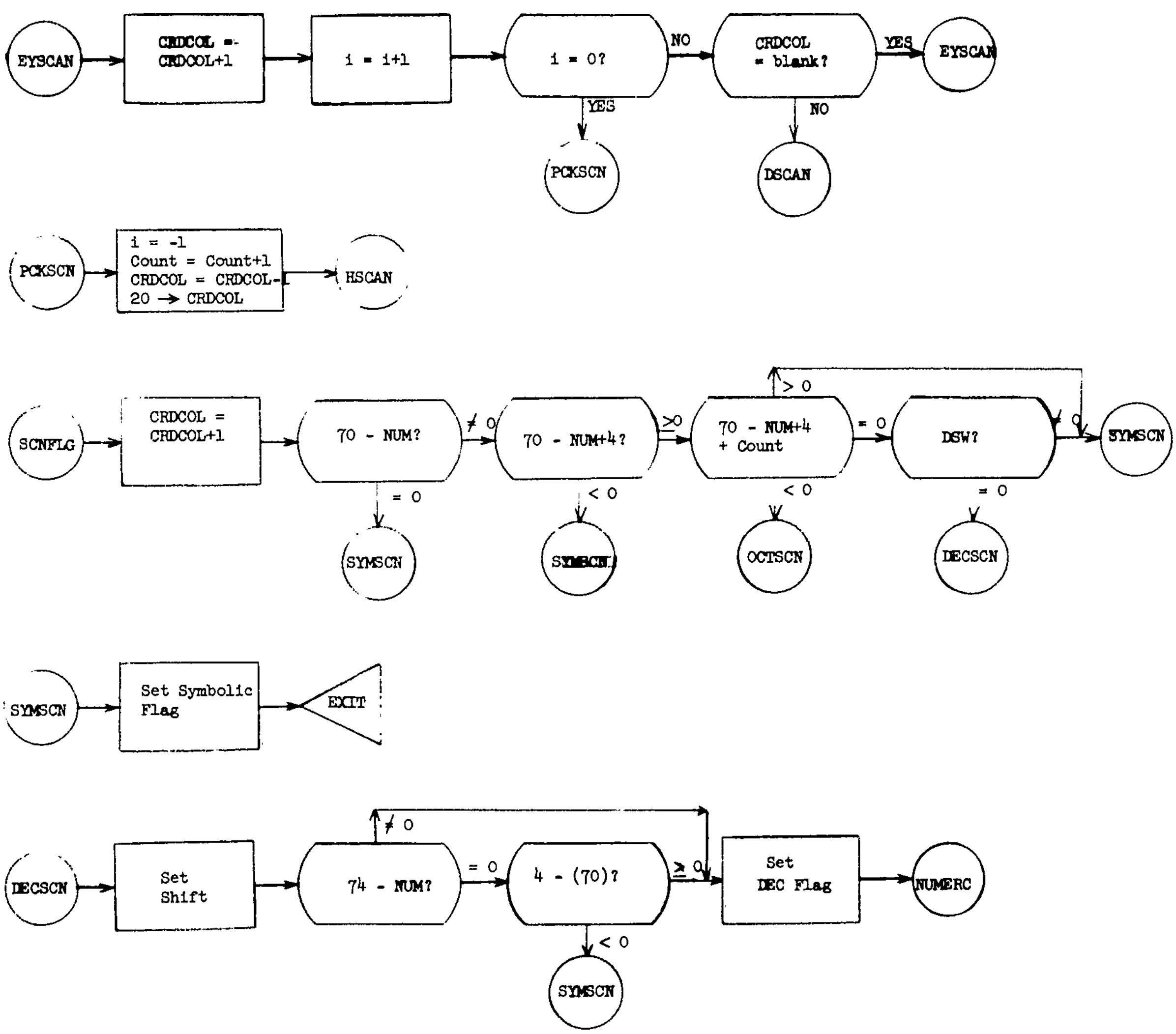

PASS2
Intermediate I/O Exist?
YES
NO
INOUT
Suppress Flag Set?
YES
NO
Set Suppress Flag
Reset Line Counter
Zero Binary Card Image
C
C
Set First Time Print Flag
Intermediate I/O Exist?
YES
NO
INTIN
REPEAT
Blank Line Image
Set Error Flags X,O,M
Illegal Op Flag Set?
NO
OPFORK
YES
STDFE
OPFORK
Op Code Switch
INDORI
EFF
BEE
ARR
TWOP
NOAD
BLOP
NUMOP
BLRSS
NED
EQUAL
ROG
RPG
ONC
REMARK
BCFLT
BEE
Set Flag For Backward Address Mode
EFF
Obtain Relative Difference
Backward Flag Set?
YES
NO
Complement
Out of Range?
NO
INDORI
YES
SERR
INDORI
NNMOP
E Term Error?
YES
SERR
Set E Error Flag
NO
Return to A
BLOP
Complement in Addr. & Addi.
SIDFE
Save E and F
NOAD
Update Line Count
SETLIN

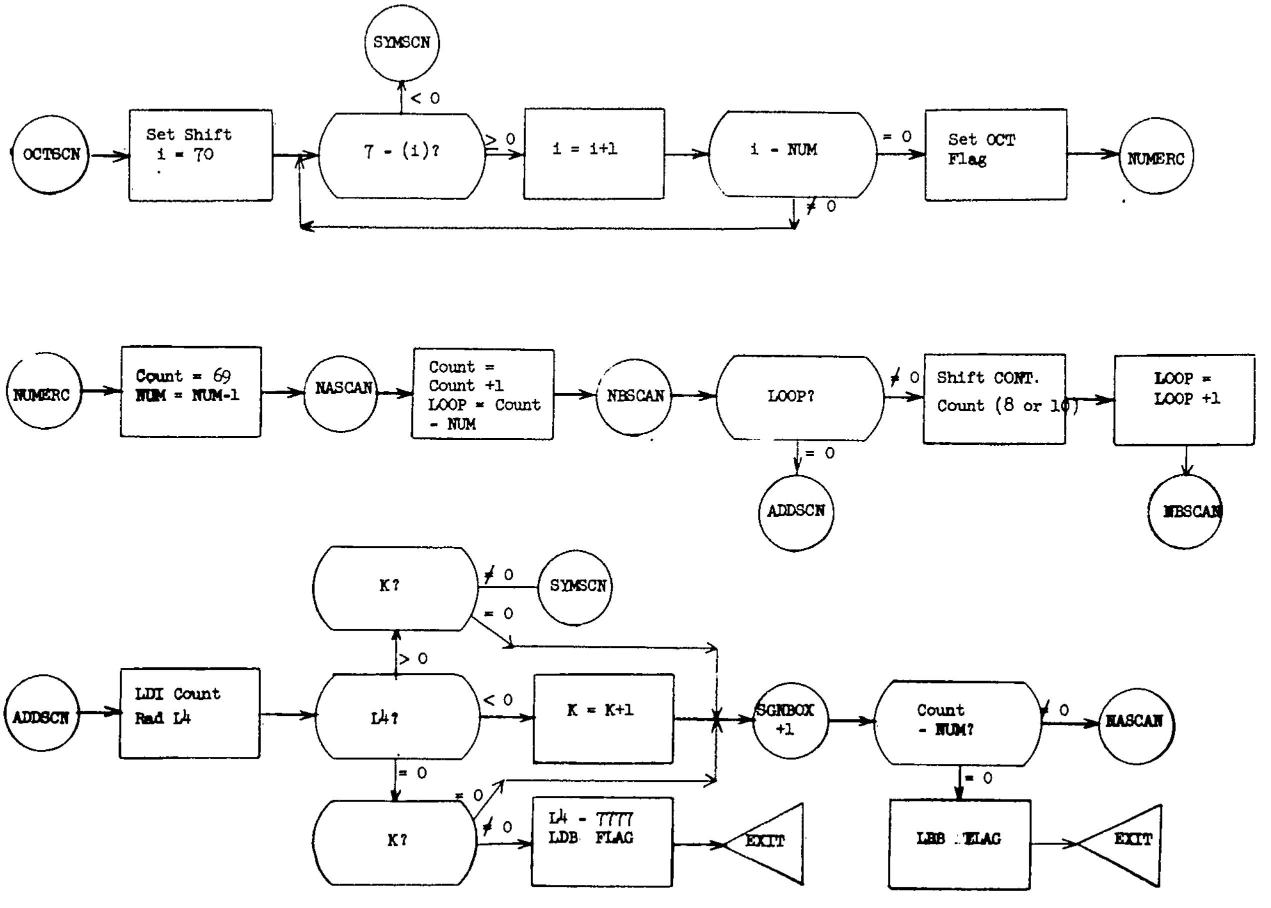

ARR
Obtain Relative Difference
In forward Range?
NO
YES
NDORI
Modify E-Term to Indicate Back
Out Of Range?
NO
NDORI
YES
SERR
TWOP
Set Relocation Bit
Update Constant = 2
NOAD
BLRSS
Update Line Count By Addr. & Addi.
SETLIN
NED
Set ENDFLG
ONC
ROG
PRG
Reset Line Count
EQUAL
In Proper Range?
NO
Set L Error Flag
SETLIN
YES
Obtain Start Address
Character Mode?
NO
Character to Words 2 at a Time
Update Constant = No. of Char. /2
NOAD
YES
Characters to Words 1 at a Time
Update Constant = No. of Char.


REMARK
BNK Card?
YES
NO
Set Flag For Bank Card
SETLIN
SETLIN
Error Flags Exist?
YES
NO
Set Error Flag In Line
Update Error Count
Octal Location to Line
Octal Contents to Line
Location Symbol to Line
STOROP
STOROP
Op Code to Line
Address Field to Line
Additive Field to Line
Remarks Field to Line
LSTOUT
IFBIN
IFBIN
Binary Output?
YES
EBW
Advance Line Counter
REPEAT
NO
NO
Bank or End Card
YES
BORE
END?
NO
YES
Error Count to A
Error Count 0?
NO
ERROR HALT
FINAL HALT
YES

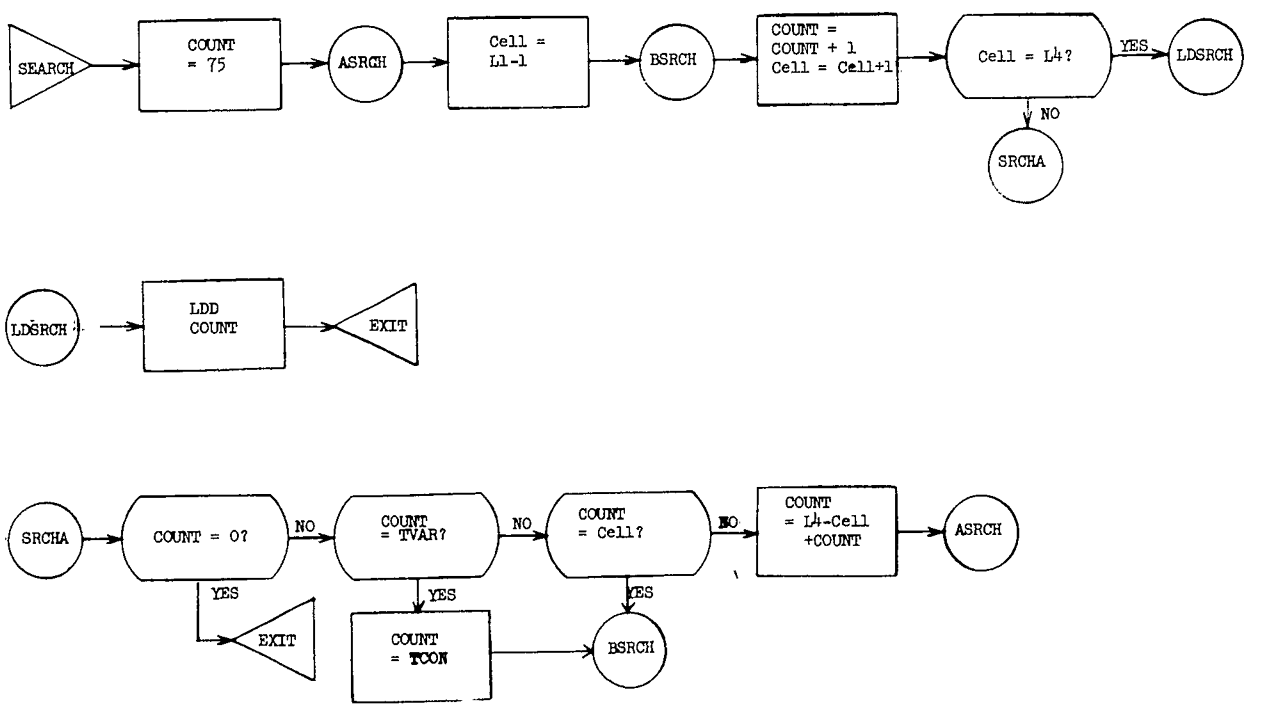


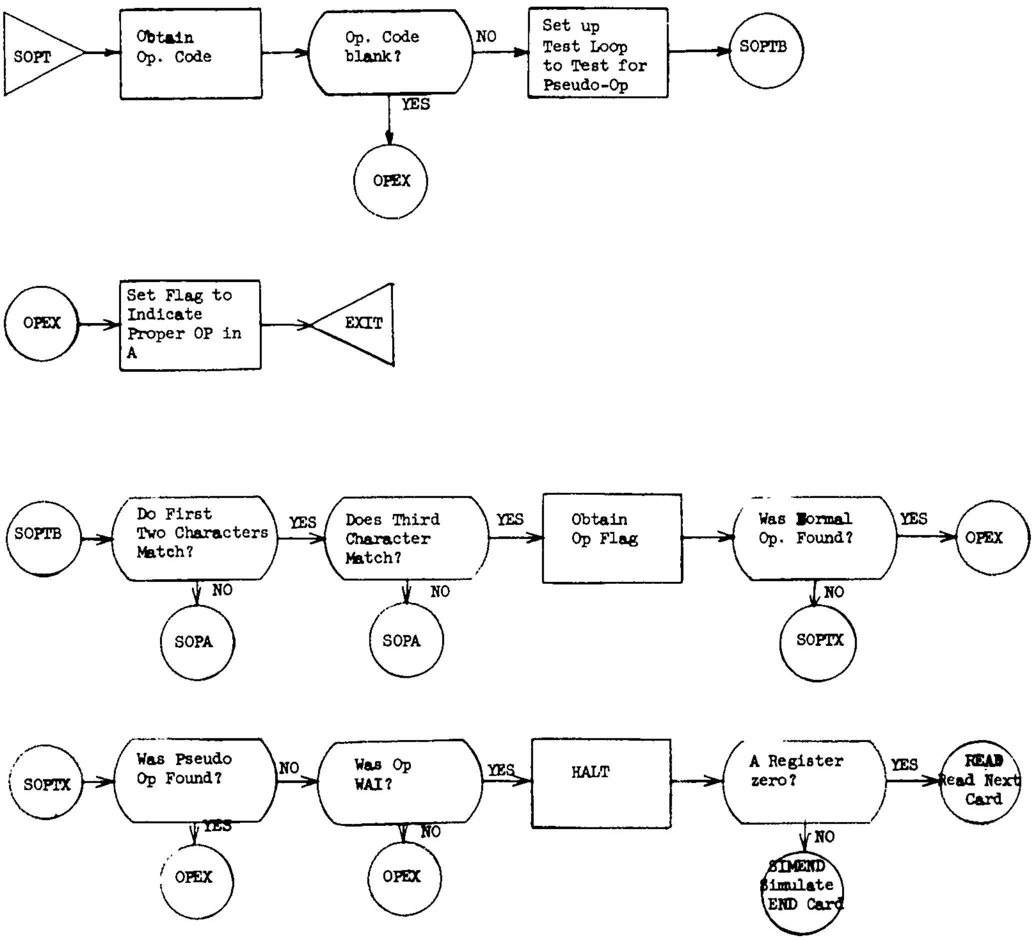
SCAN
i = 7, K = -6
NUM = 70, COUNT
= -1, j = L1,
DSW = -1, Flg = 0
BSCAN
CRDCOL
= blank ?
YES
NO
CSCAN
CRDCOL
= CRDCOL+1
i = i+1
i = 0?
NO
YES
EXIT
CSCAN
CRDCOL
= + or -?
YES
NO
DSCAN
Set
Sign Flag
CSCAN1
CRDCOL =
CRDCOL+1
i = i+1
i = 0?
NO
YES
EXIT
CRDCOL
= blank ?
YES
NO
DSCAN
DSCAN
CRDCOL
- 12?
= 0
< 0
> 0
Add 12
STI NUM
NUM = NUM+1
ESCAN1
DSW
= 0?
YES
NO
HSCAN
DSW
= DSW+1
HSCAN
ESCAN
CRDCOL
= D?
NO
YES
HSCAN
Shift Replace
2 - Count
Switch
Test Switch
for + or -
+
-
LDI CRDCOL
SHA 11
STI J
LDI CRDCOL
HWI J
J = J+1
K = K+1
K = 0?
NO
YES
EYSCAN
SCNFLG

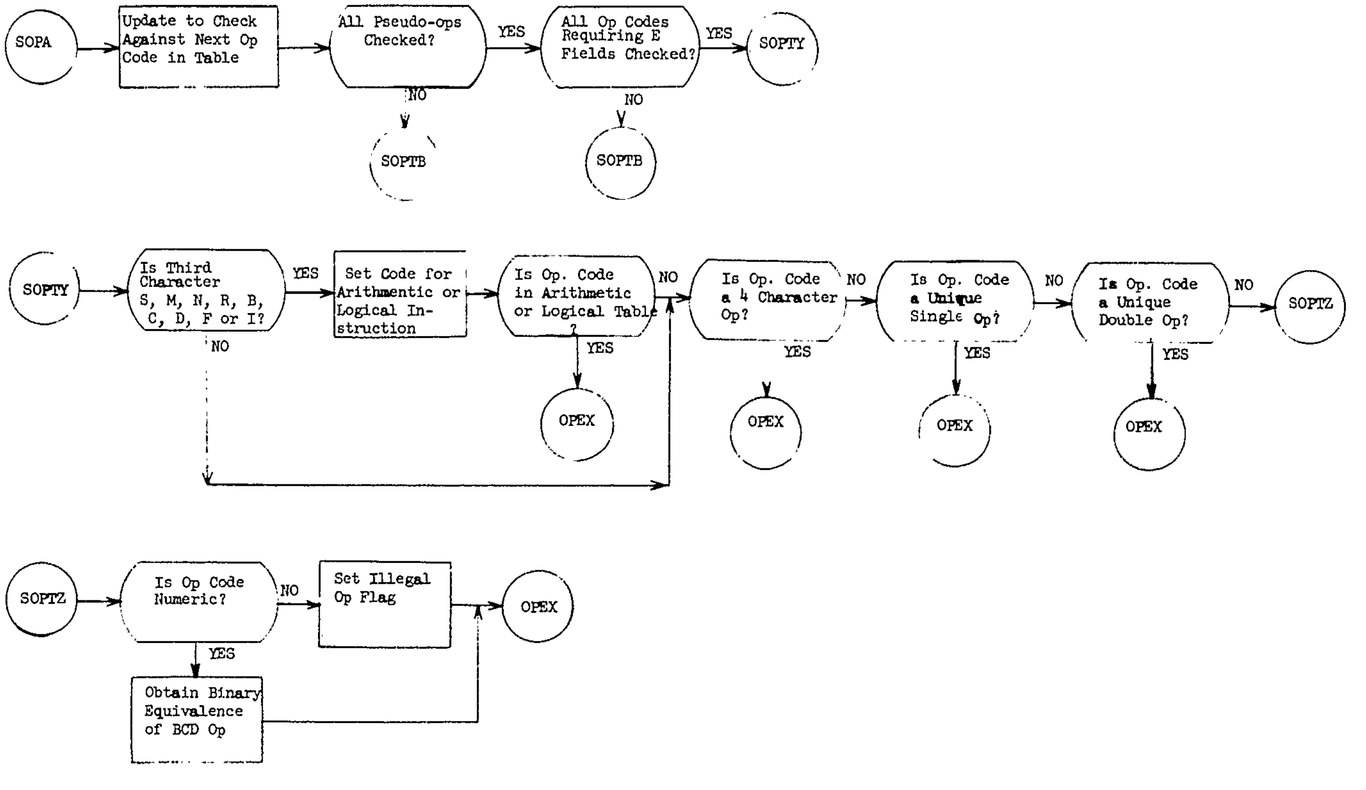

EYSCAN
CRDCOL = CRDCOL+1
i = i+1
i = 0?
NO
YES
CRDCOL = blank?
YES
EYSCAN
NO
PCKSCN
DSCAN
PCKSCN
i = -1
Count = Count+1
CRDCOL = CRDCOL-1
20 → CRDCOL
HSCAN
SCNFLG
CRDCOL = CRDCOL+1
70 - NUM?
≠ 0
= 0
SYMSCN
70 - NUM+4?
≥0
< 0
SYMSCN
70 - NUM+4 + Count
> 0
= 0
< 0
OCTSCN
DSW?
≠ 0
= 0
DECSCN
SYMSCN
SYMSCN
Set Symbolic Flag
EXIT
DECSCN
Set Shift
74 - NUM?
≠ 0
= 0
4 - (70)?
≥ 0
< 0
SYMSCN
Set DEC Flag
NUMERC

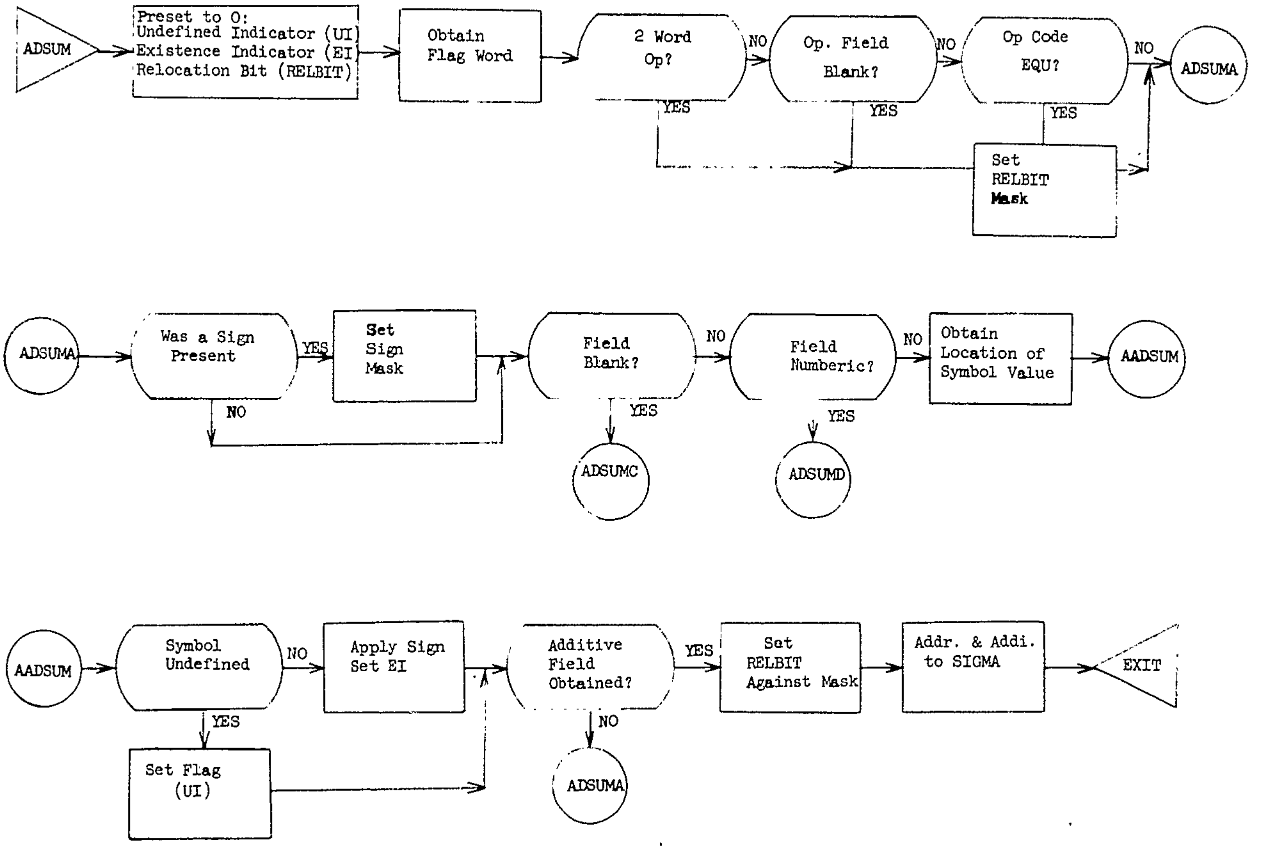


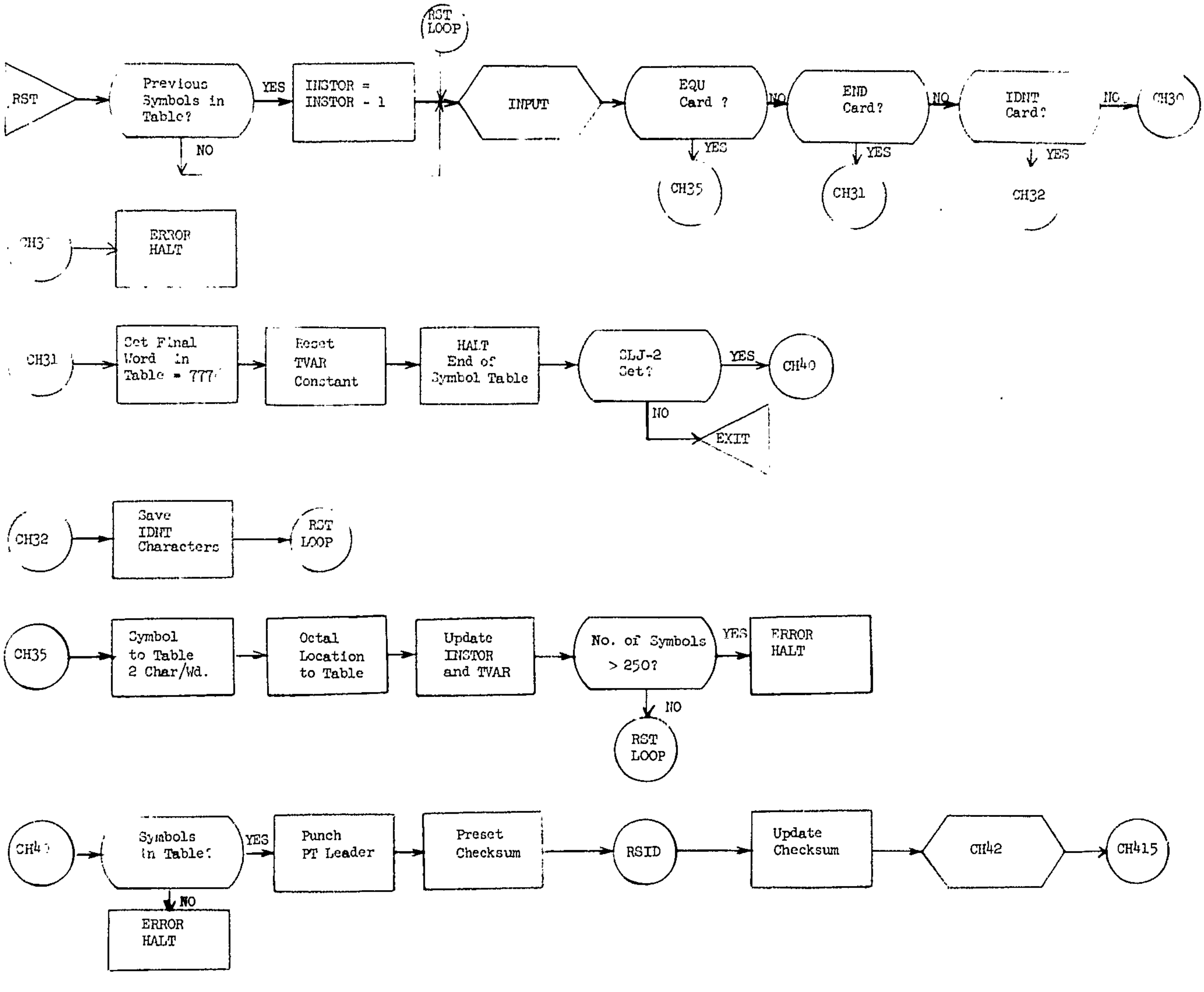
SYMSCN
< 0
OCTSCN
Set Shift
i = 70
7 - (i)?
> 0
i = i+1
i - NUM
= 0
≠ 0
Set OCT
Flag
NUMERC
NUMERC
Count = 69
NUM = NUM-1
NASCAN
Count =
Count +1
LOOP = Count
- NUM
NBSCAN
LOOP?
≠ 0
= 0
Shift CONT.
Count (8 or 10)
LOOP =
LOOP +1
ADDSCN
NBSCAN
K?
≠ 0
= 0
SYMSCN
> 0
ADDSCN
LDI Count
Rad L4
L4?
< 0
K = K+1
SGNBOX
+1
Count
- NUM?
≠ 0
NASCAN
= 0
= 0
K?
≠ 0
L4 - 7777
LDB FLAG
EXIT
= 0
LDB FLAG
EXIT

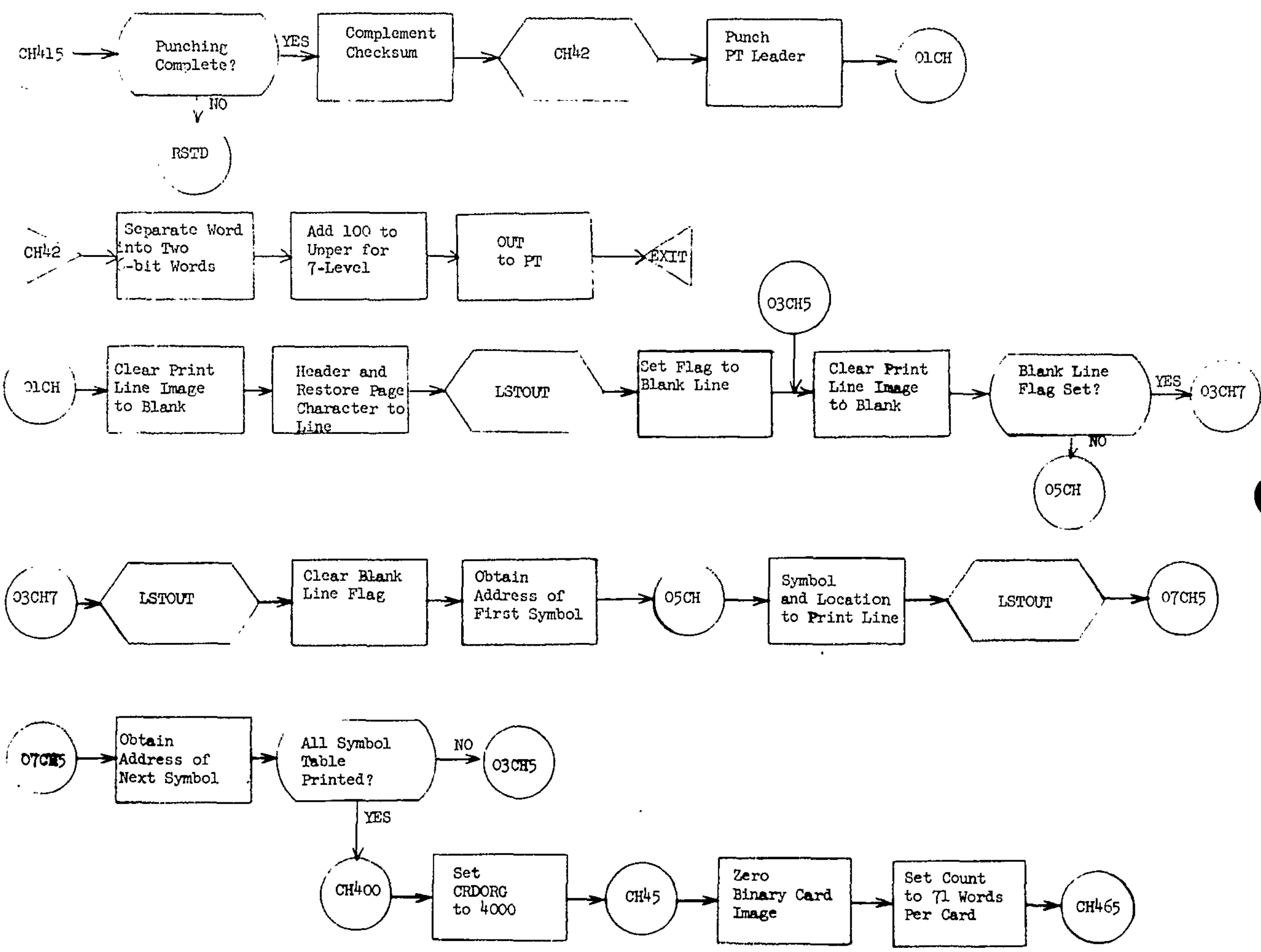

SEARCH
COUNT = 75
ASRCH
Cell = L1-1
BSRCH
COUNT = COUNT + 1
Cell = Cell+1
Cell = L4?
YES
LDSRCH
NO
SRCHA
LDSRCH
LDD COUNT
EXIT
SRCHA
COUNT = 0?
NO
YES
EXIT
COUNT = TVAR?
NO
YES
COUNT = TCON
COUNT = Cell?
NO
YES
BSRCH
COUNT = L4-Cell +COUNT
ASRCH

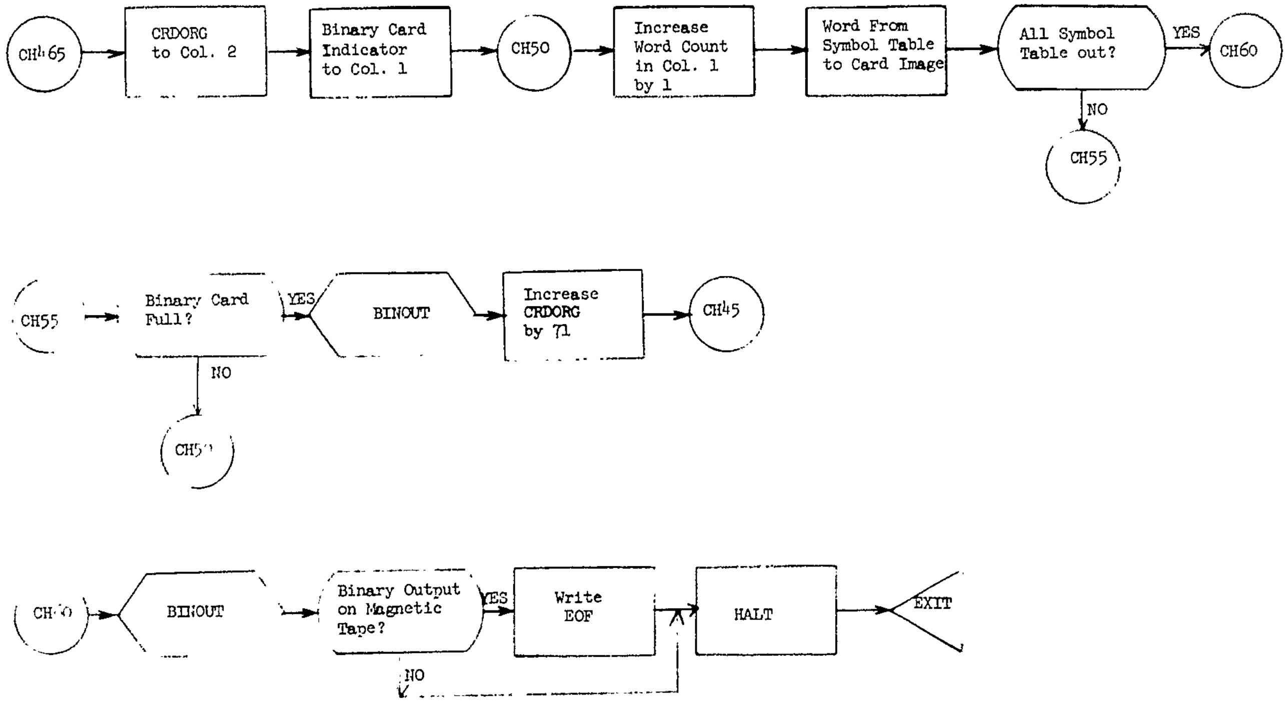

SOPT
Obtain Op. Code
Op. Code blank?
NO
YES
Set up Test Loop to Test for Pseudo-Op
SOPTB
OPEX
OPEX
Set Flag to Indicate Proper OP in A
EXIT
SOPTB
Do First Two Characters Match?
YES
NO
SOPA
Does Third Character Match?
YES
NO
SOPA
Obtain Op Flag
Was Normal Op. Found?
YES
NO
OPEX
SOPTX
SOPTX
Was Pseudo Op Found?
NO
YES
OPEX
Was Op WAI?
YES
NO
OPEX
HALT
A Register zero?
YES
NO
READ Read Next Card
SIMEND Simulate END Card

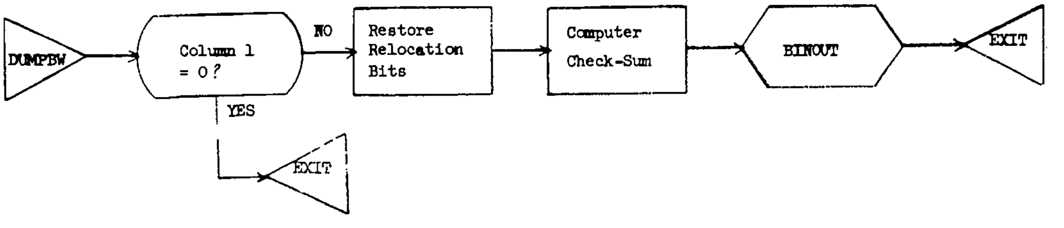

SOPA
Update to Check Against Next Op Code in Table
All Pseudo-ops Checked?
YES
NO
SOPTB
All Op Codes Requiring E Fields Checked?
YES
NO
SOPTB
SOPTY
SOPTY
Is Third Character S, M, N, R, B, C, D, F or I?
YES
NO
Set Code for Arithmentic or Logical Instruction
Is Op. Code in Arithmetic or Logical Table?
NO
YES
OPEX
Is Op. Code a 4 Character Op?
NO
YES
OPEX
Is Op. Code a Unique Single Op?
NO
YES
OPEX
Is Op. Code a Unique Double Op?
NO
YES
OPEX
SOPTZ
SOPTZ
Is Op Code Numeric?
NO
Set Illegal Op Flag
OPEX
YES
Obtain Binary Equivalence of BCD Op

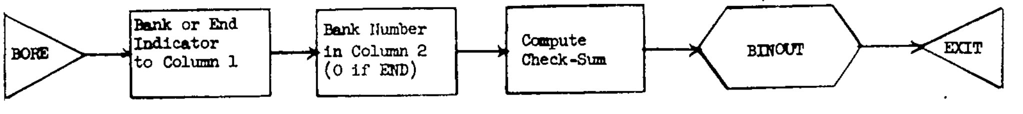

ADSUM
Preset to 0: Undefined Indicator (UI) Existence Indicator (EI) Relocation Bit (RELBIT)
Obtain Flag Word
2 Word Op?
NO
YES
Op. Field Blank?
NO
YES
Op Code EQU?
NO
YES
ADSUMA
Set RELBIT Mask
ADSUMA
Was a Sign Present
YES
NO
Set Sign Mask
Field Blank?
NO
YES
ADSUMC
Field Numberic?
NO
YES
ADSUMD
Obtain Location of Symbol Value
AADSUM
AADSUM
Symbol Undefined
NO
YES
Apply Sign Set EI
Set Flag (UI)
Additive Field Obtained?
YES
NO
ADSUMA
Set RELBIT Against Mask
Addr. & Addi. to SIGMA
EXIT


RST
Previous Symbols in Table?
YES
NO
INSTOR = INSTOR - 1
RST LOOP
INPUT
EQU Card ?
NO
YES
CH35
END Card?
NO
YES
CH31
IDNT Card?
NO
YES
CH32
ERROR HALT
CH31
Set Final Word in Table = 777
Reset TVAR Constant
HALT End of Symbol Table
SLJ-2 Set?
YES
CH40
NO
EXIT
CH32
Save IDNT Characters
RST LOOP
CH35
Symbol to Table 2 Char/Wd.
Octal Location to Table
Update INSTOR and TVAR
No. of Symbols > 250?
YES
ERROR HALT
NO
RST LOOP
Symbols in Table?
YES
Punch PT Leader
Preset Checksum
RSID
Update Checksum
CH42
CH415
NO
ERROR HALT

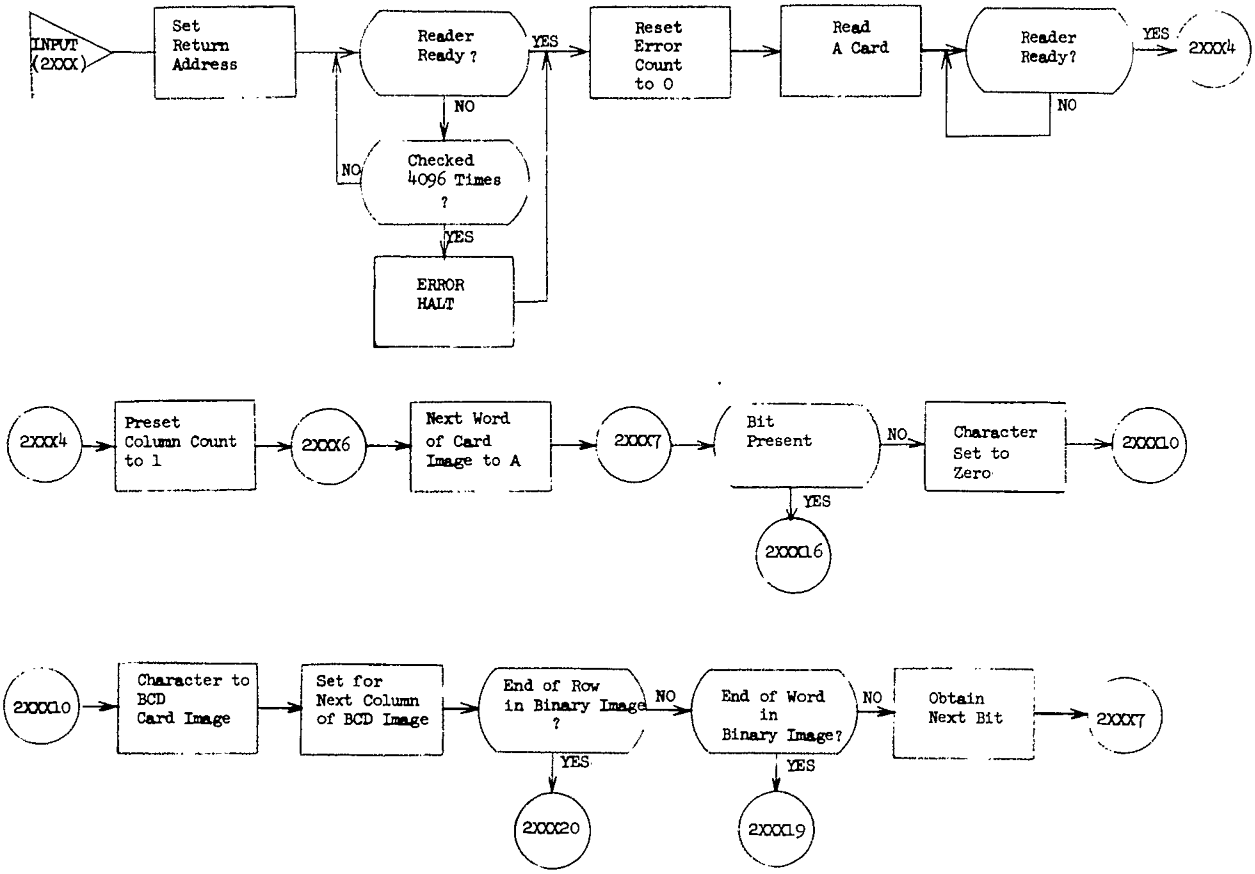

CH415
Punching Complete?
YES
NO
Complement Checksum
CH42
Punch PT Leader
01CH
RSTD
CH42
Separate Word into Two -bit Words
Add 100 to Upper for 7-Level
OUT to PT
EXIT
03CH5
01CH
Clear Print Line Image to Blank
Header and Restore Page Character to Line
LSTOUT
Set Flag to Blank Line
Clear Print Line Image to Blank
Blank Line Flag Set?
YES
03CH7
NO
05CH
03CH7
LSTOUT
Clear Blank Line Flag
Obtain Address of First Symbol
05CH
Symbol and Location to Print Line
LSTOUT
07CH5
07CH5
Obtain Address of Next Symbol
All Symbol Table Printed?
NO
03CH5
YES
CH400
Set CRDORG to 4000
CH45
Zero Binary Card Image
Set Count to 71 Words Per Card
CH465

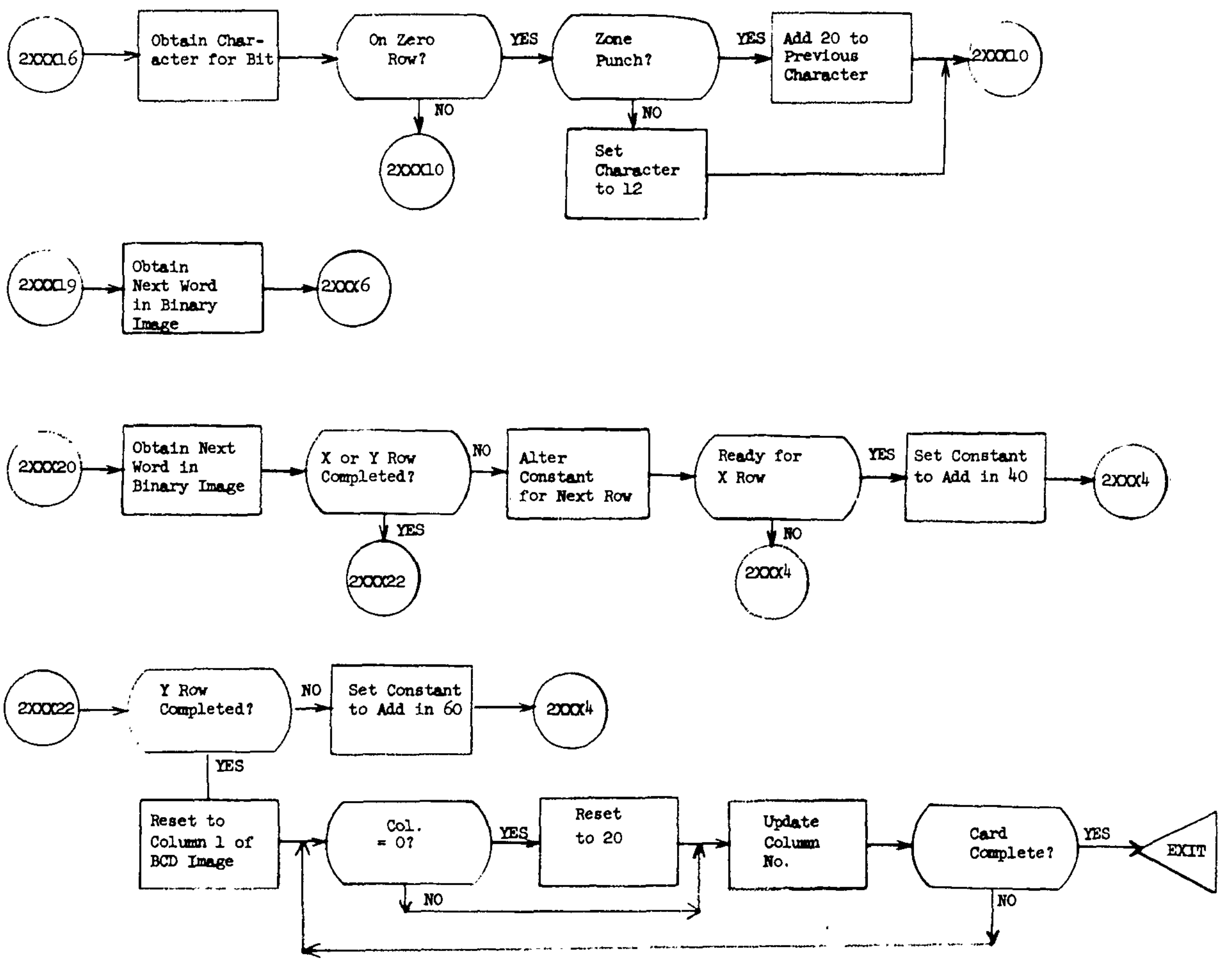

CH45
CRDORG to Col. 2
Binary Card Indicator to Col. 1
CH50
Increase Word Count in Col. 1 by 1
Word From Symbol Table to Card Image
All Symbol Table out?
YES
CH60
NO
CH55
CH55
Binary Card Full?
YES
BINOUT
Increase CRDORG by 71
CH45
NO
BINOUT
Binary Output on Magnetic Tape?
YES
Write EOF
HALT
EXIT
NO

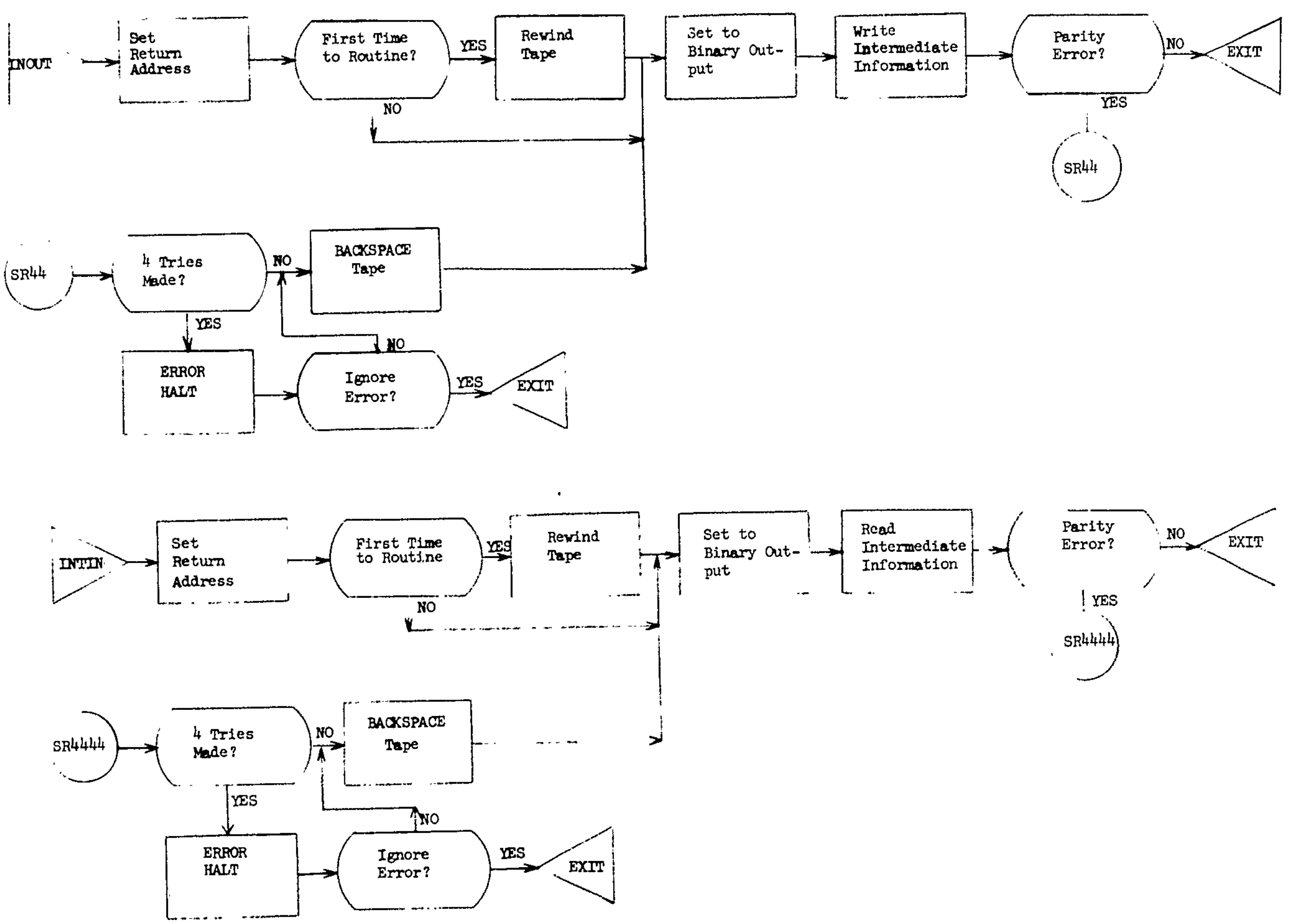

DUMPBW
Column 1
= 0?
NO
YES
Restore
Relocation
Bits
Computer
Check-Sum
BINOUT
EXIT
EXIT


BORE
Bank or End
Indicator
to Column 1
Bank Number
in Column 2
(0 if END)
Compute
Check-Sum
BINOUT
EXIT

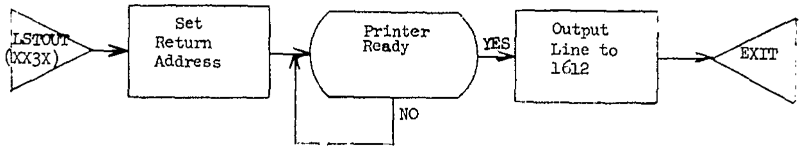
VERSION A


INPUT (2XXX)
Set Return Address
Reader Ready ?
YES
NO
Checked 4096 Times ?
NO
YES
ERROR HALT
Reset Error Count to 0
Read A Card
Reader Ready?
YES
NO
2XXX4
2XXX4
Preset Column Count to 1
2XXX6
Next Word of Card Image to A
2XXX7
Bit Present
NO
YES
Character Set to Zero
2XXX10
2XXX16
2XXX10
Character to BCD Card Image
Set for Next Column of BCD Image
End of Row in Binary Image ?
NO
YES
2XXX20
End of Word in Binary Image?
NO
YES
2XXX19
Obtain Next Bit
2XXX7

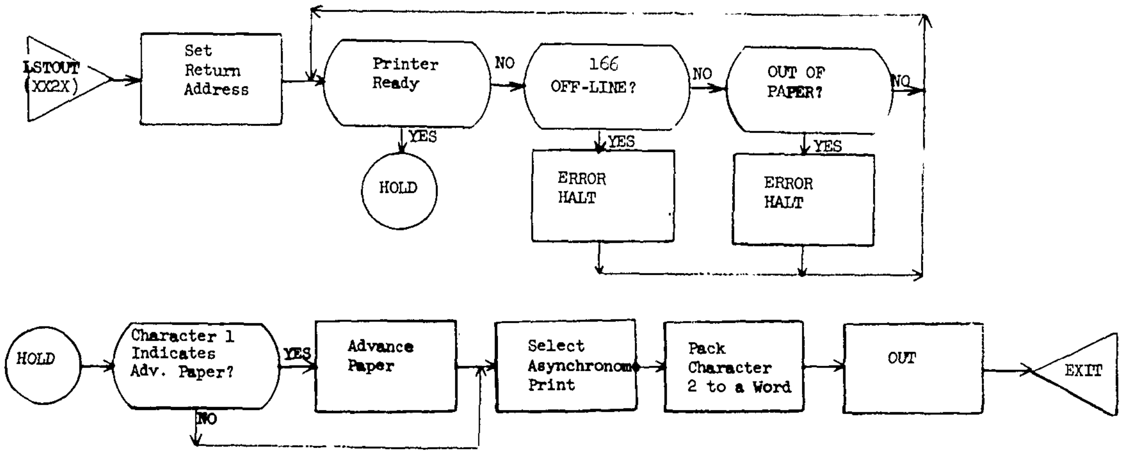

2XXX16
Obtain Character for Bit
On Zero Row?
YES
NO
2XXX10
Zone Punch?
YES
NO
Add 20 to Previous Character
2XXX10
Set Character to 12
2XXX19
Obtain Next Word in Binary Image
2XXX6
2XXX20
Obtain Next Word in Binary Image
X or Y Row Completed?
NO
YES
2XXX22
Alter Constant for Next Row
Ready for X Row
YES
NO
2XXX4
Set Constant to Add in 40
2XXX4
2XXX22
Y Row Completed?
NO
Set Constant to Add in 60
2XXX4
YES
Reset to Column 1 of BCD Image
Col. = 0?
YES
NO
Reset to 20
Update Column No.
Card Complete?
YES
NO
EXIT


INOUT
Set Return Address
First Time to Routine?
YES
NO
Rewind Tape
Set to Binary Out-put
Write Intermediate Information
Parity Error?
NO
EXIT
YES
SR44
SR44
4 Tries Made?
NO
BACKSPACE Tape
YES
NO
ERROR HALT
Ignore Error?
YES
EXIT
INTIN
Set Return Address
First Time to Routine?
YES
NO
Rewind Tape
Set to Binary Out-put
Read Intermediate Information
Parity Error?
NO
EXIT
YES
SR4444
SR4444
4 Tries Made?
NO
BACKSPACE Tape
YES
NO
ERROR HALT
Ignore Error?
YES
EXIT

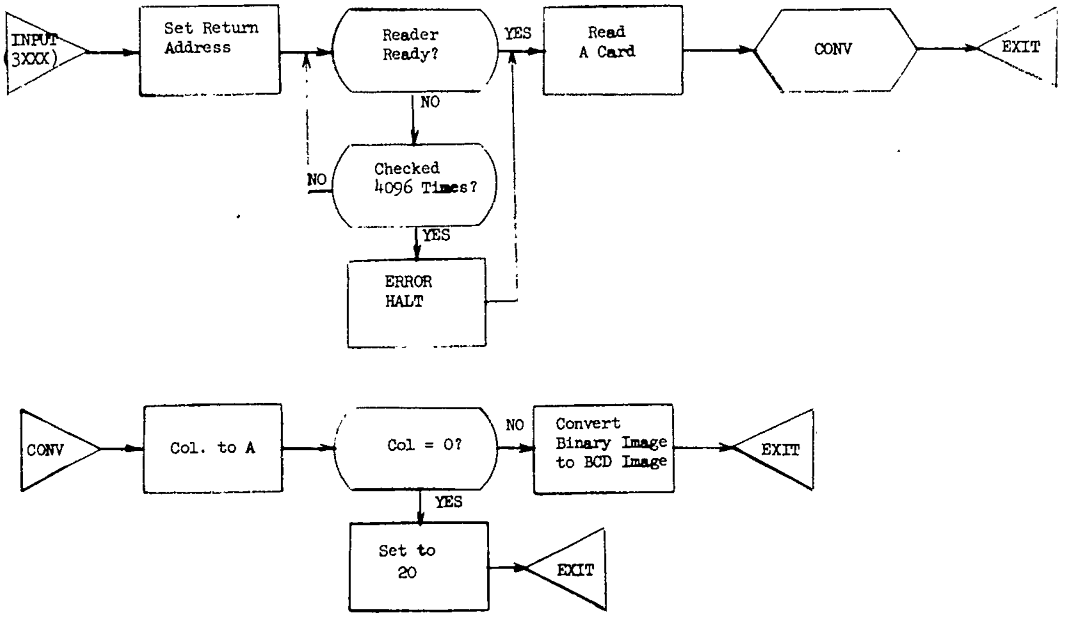
VERSION A



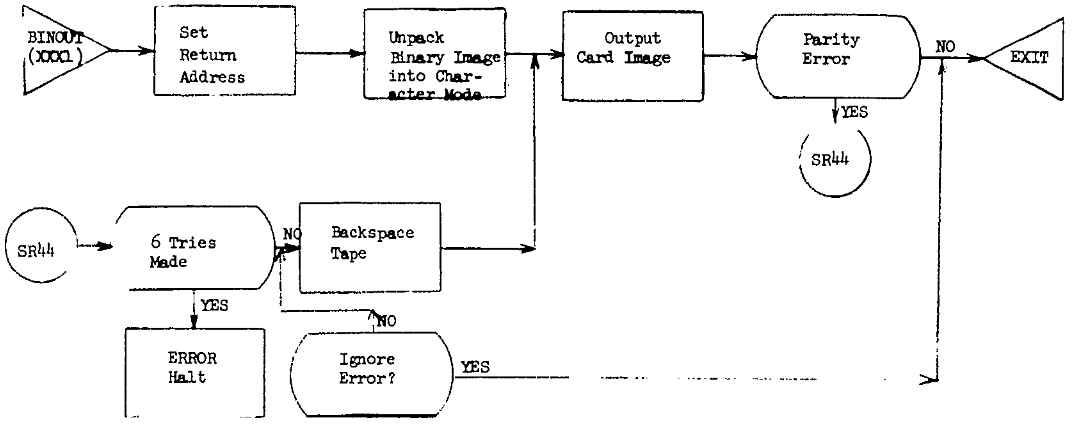
VERSION B



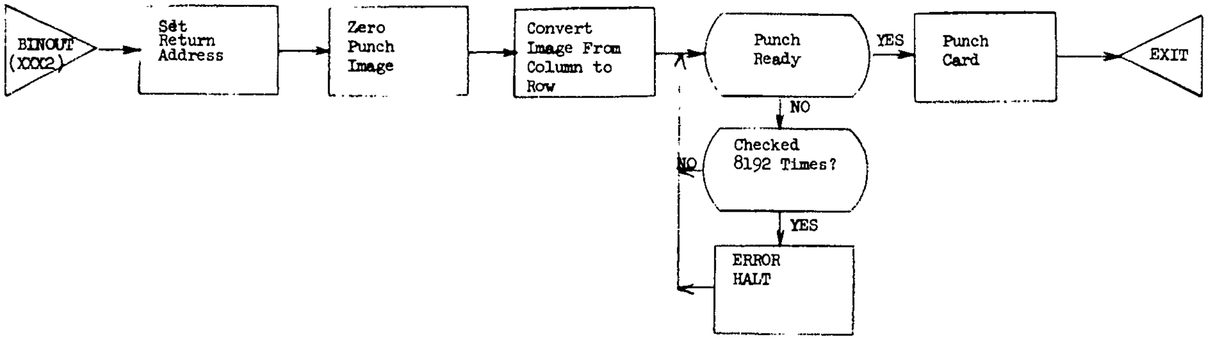
VERSION B


INPUT (3XXX)
Set Return Address
Reader Ready?
YES
NO
Read A Card
CONV
EXIT
Checked 4096 Times?
NO
YES
ERROR HALT
CONV
Col. to A
Col = 0?
NO
Convert Binary Image to BCD Image
EXIT
YES
Set to 20
EXIT

VERSION B



VERSION A



APPENDIX A

MNEMONIC CODES RECOGNIZED BY OSAS-A

Where X or XX = Any Octal digit or digits
XXXX = Octal Operand or EF code
YYYY = Octal Address

Mnemonic	Name	F	E	G	Type*
ACJ	Set D, I, and R Bank Control and Jump	00	7X		N
ADB	Add Backward	33	XX		N
ADC	Add Constant	32	00	XXXX	N
ADD	Add Direct	30	XX		N
ADF	Add Forward	32	XX		N
ADI	Add Indirect	31	XX		
ADM	Add Memory	31	00	YYYY	N
ADN	Add No Address	06	XX		N
ADR	Add Relative	3X	XX		S
ADS	Add Specific	33	00		N
AOB	Replace Add One Backward	57	XX		N
AOC	Replace Add One Constant	56	00	XXXX	N
AOD	Replace Add One Direct	54	XX		N
AOF	Replace Add One Forward	56	XX		N
AOI	Replace Add One Indirect	55	XX		N
AOM	Replace Add One Memory	55	00	YYYY	N
AOR	Replace Add One Relative	5X	XX		S
AOS	Replace Add One Specific	57	00		N
ATE	A to Buffer Entrance Register	01	05	YYYY	N
ATX	A to Buffer Exit Register	01	06	YYYY	N
BCD	2 BCD Characters/word				P
BCDR	1 BCD Character/word				P
BLR	Block Storage Reserve				P
BLS	Block Store	01	00	YYYY	N

* Normal Codes = N, pseudo-ops = P, and special relative codes = S.

Mnemonic	Name	F	E	G	Type*
BNKX	Generate Binary Bank Card				
BSS	Block Storage Reserve				P
CBC	Clear Buffer Controls	01	04		N
CIL	Clear Interrupt Lockout	01	20		N
CTA	Bank Controls to A	01	30		N
DRJ	Set D and R Bank Control and Jump	00	5X		N
END	End				P
ETA	Buffer Entrance Register to A	01	07		N
EQU	Equality				P
ERR	Error Stop	00	00		N
EXC	External Function Constant	75	00	XXXX	N
EXF	External Function Forward	75	XX		N
FLX	2 Flex Characters/word				P
FLXR	1 Flex Character/word				P
HLT	Halt	77 77	00 77		N
HWI	Half Write Indirect	76	XX		N
IBI	Initiate Buffer Input	72	00	YYYY	N
IBO	Initiate Buffer Output	73	00	YYYY	N
INA	Input to A	76	00		N
INP	Input	72	XX	YYYY	N
IRJ	Set I and R Bank Control and Jump	00	3X		N
JFI	Jump Forward Indirect	71	XX		N
JPI	Jump Indirect	70	XX		N
JPR	Return Jump	71	00	YYYY	N

* Normal codes = N, pseudo-ops = P, and special relative codes = S.

Mnemonic	Name	F	E	G	Type*
LCB	Load Complement Backward	27	XX		N
LCC	Load Complement Constant	26	00	XXXX	N
LCD	Load Complement Direct	24	XX		N
LCF	Load Complement Forward	26	XX		N
LCI	Load Complement Indirect	25	XX		N
LCM	Load Complement Memory	25	00	YYYY	N
LCN	Load Complement No Address	05	XX		N
LCR	Load Complement Relative	2X	XX		S
LCS	Load Complement Specific	27	00		N
LDB	Load Backward	23	XX		N
LDC	Load Constant	22	00	XXXX	N
LDD	Load Direct	20	XX		N
LDF	Load Forward	22	XX		N
LDI	Load Indirect	21	XX		N
LDM	Load Memory	21	00	YYYY	N
LDN	Load No Address	04	XX		N
LDR	Load Relative	2X	XX		S
LDS	Load Specific	23	00		N
LPB	Logical Product Backward	13	XX		N
LPC	Logical Product Constant	12	00	XXXX	N
LPD	Logical Pr[illegible] Direct	10	XX		N
LPF	Logical Product Forward	12	XX		N
LPI	Logical Product Indirect	11	XX		N
LPM	Logical Product Memory	11	00	YYYY	N
LPN	Logical Product No Address	02	XX		N
LPR	Logical Product Relative	1X	XX		S
LPS	Logical Product Specific	13	00		N
LSB	Logical Sum Backward (exclusive "or")	17	XX		N
LSD	Logical Sum Direct (exclusive "or")	14	XX		N

* Normal codes = N, pseudo-ops = P, and special relative codes = S.

Mnemonic	Name	F	E	G	Type*
LSF	Logical Sum Forward (exclusive "or)	16	XX		N
LSI	Logical Sum Indirect (exclusive "or")	15	XX		N
LSN	Logical Sum No Address (exclusive "or")	03	XX		N
LSR	Logical Sum Relative (exclusive "or")	1X	XX		S
LS1	Left Shift One	01	02		N
LS2	Left Shift Two	01	03		N
LS3	Left Shift Three	01	10		N
LS6	Left Shift Six	01	11		N
MUH	Multiply A by One Hundred	01	13		N
MUT	Multiply A by Ten	01	12		N
NJB	Negative Jump Backward	67	XX		N
NJF	Negative Jump Forward	63	XX		N
NJR	Negative Jump Relative	6X	XX		S
NOP	No Operation	00	0X		N
NZB	Non-Zero Jump Backward	65	XX		N
NZF	Non-Zero Jump Forward	61	XX		N
NZR	Non-Zero Jump Relative	6X	XX		S
OTA	Output from A	76	77		N
OTN	Output No Address	74	XX		N
OUT	Output	73	XX	YYYY	N
ORG	Origin				P
PJB	Positive Jump Backward	66	XX		N
PJF	Positive Jump Forward	62	XX		N
PJR	Positive Jump Relative	6X	XX		S
PRG	Program				P
PTA	Transfer P to A	01	01		N
RAB	Replace Add Backward	53	XX		N
RAC	Replace Add Constant	52	00	XXXX	N

* Normal codes = N, pseudo-ops = P, and special relative codes = S.

Mnemonic	Name	F	E	G	Type*
RAD	Replace Add Direct	50	XX		N
RAF	Replace Add Forward	52	XX		N
RAI	Replace Add Indirect	51	XX		N
RAM	Replace Add Memory	51	00	YYYY	N
RAR	Replace Add Relative	5X	XX		S
RAS	Replace Add Specific	53	00		N
REM	Remarks				P
RS1	Right Shift One	01	14		N
RS2	Right Shift Two	01	15		N
SBB	Subtract Backward	37	XX		N
SBC	Subtract Constant	36	00	XXXX	N
SBD	Subtract Direct	34	XX		N
SBF	Subtract Forward	36	XX		N
SBI	Subtract Indirect	35	XX		N
SBM	Subtract Memory	35	00	YYYY	N
SBN	Subtract No Address	07	XX		N
SBR	Subtract Relative	3X	XX		S
SBS	Subtract Specific	37	00		N
SBU	Set Buffer Bank Control	01	4X		N
SCB	Selective Complement Backward	17	XX		N
SCC	Selective Complement Constant	16	00	XXXX	N
SCD	Selective Complement Direct	14	XX		N
SCF	Selective Complement Forward	16	XX		N
SCI	Selective Complement Indirect	15	XX		N
SCM	Selective Complement Memory	15	00	YYYY	N
SCN	Selective Complement No Address	03	XX		N
SCR	Selective Complement Relative	1X	XX		S
SCS	Selective Complement Specific	17	00		N
SDC	Set Direct Bank Control	00	4X		N

* Normal codes - N, pseudo-ops = P, and special relative codes = S.

Mnemonic	Name	F	E	G	Type*
SHA	Shift A Left	01			N
SIC	Set Indirect Bank Control	00	2X		N
SID	Set Indirect and Direct Bank Control	00	6X		N
SLJ	Selective Jump	77	X0	YYYY	N
SLS	Selective Stop	77	0X		N
SJS	Selective Stop and Jump	77	XX	YYYY	N
SRB	Shift Replace Backward	47	XX		N
SRC	Shift Replace Constant	46	00	XXXX	N
SRD	Shift Replace Direct	44	XX		N
SRF	Shift Replace Forward	46	XX		N
SRI	Shift Replace Indirect	45	XX		N
SRJ	Set Relative Bank Control and Jump	00	1X		N
SRM	Shift Replace Memory	45	00	YYYY	N
SRR	Shift Replace Relative	4X	XX		S
SRS	Shift Replace Specific	47	00		N
STB	Store Backward	43	XX		N
STC	Store Constant	42	00	XXXX	N
STD	Store Direct	40	XX		N
STE	BER to Location 6X, A to BER	01	6X		N
STF	Store Forward	42	XX		N
STI	Store Indirect	41	XX		N
STM	Store Memory	41	00	YYYY	N
STP	P to Location 5X	01	5X		N
STR	Store Relative	4X	XX		S
STS	Store Specific	43	00		N
SUPA	Suppress Listable Output				P
SUPB	Suppress Binary Output				P

* Normal codes = N, pseudo-ops = P, and special relative codes = S.

Mnemonic	Name	F	E	G	Type*
WAI	Wait				P
ZJB	Zero Jump Backward	64	XX		N
ZJF	Zero Jump Forward	64	XX		N
ZJR	Zero Jump Relative	6X	XX		S

* Normal codes = N, pseudo-ops = P, and special relative codes = S.

APPENDIX B

EXAMPLE

LOC	SYMBOL	OP	ADDRESS	ADDITIVE
		ORG	1000	
*1000	OMEGA	LDD	ALPHA	
*1001		STD	BETA	
*1002		AOD	DELTA	
*1003		RAD	DELTA	+1
		CON	43	
0043	ALPHA		35	
0044	BETA			
0045	DELTA			
		PRG	(an ORG here with blank AA fields would set the LDF location to 0. An ORG with AA fields OMEGA + 4 would set the LDF location to 1004.)	
*1004		LDF	** THETA	
*1005		STM	** STORE	
*1006 + 1007		JPR	** OMEGA	
*1010	THETA			
*1011	STORE			
		END		

* Will be incremented by a relocation constant

** Will be modified by a relocation constant

APPENDIX C

SYMBOL TABLE TEST

ABCDEF	1035
BCDEFG	7101
CDEFGH	3264
DEFGHI	3256
EFGHIJ	3364
GHIJKL	7671
HIJKLM	7672
IJKLMN	7673
ZYXWV	4052
EDCBA	0440
D456	2002
E567	0110
F678	0207
A	7100
B	3620
C	6021
D	4013
JKLMNO	0000
KLMNOP	0031
LMNOPQ	0026
MNOPQR	0030
NOPQRS	0027
OPQRST	0037
PQRSTU	0042
QRSTUV	0045
RSTUVW	0100
STUVWX	7500
TUVWXY	4102
UVWXYZ	7720
VWXYZA	0111
WXYZAB	0476
XYZABC	4034
YZABCD	0501
ZABCDE	4033
A123	6110
B234	7100
C345	3676
G789	6102
H890	7700
I901	0405
J012	4004
12K34	0400
23L45	4003
34M56	4006
45N67	4051
56O78	4032
67P89	0501
78Q90	4057
89R01	3223
90S12	4005
01T23	2067
YXWVU	0140
XWVUT	0060
WVUTS	0407
VUTSR	4077
UTSRQ	0400
TSRQP	4177
SRQPO	5477
RQPON	0725
QPONM	6504
PONML	0101
ONMLK	7062
NMLKJ	5011
KJIHG	4061
JIHGF	0101
MLKJI	2200
LKJIH	7660
IHGFE	7050
HGFED	4070
GFEDC	0714
FEDCB	6303
DCBAZ	5010
CBAZY	2070
BAZYX	0203
AZYXW	6035
H	2513

```
0100  0602  START   ADN   2
0101  4210          STF   RETURN
0102  2210          LDF   ALPHA
0103  3100          ADM   BCDEFG
0104  7101
0105  1600          SCC   DEFGHI
0106  3256
0107  4204          STF   BETA
0110  7101          JFI   1
0111  0000  RETURN
0112  1324  ALPHA         1324
0113  0000  BETA
      0000          END
```

DISTRIBUTION LIST

External

Space Systems Division
(Contracting Agency)
Major C. R. Bond (SSOCD)

6594th Aerospace Test Wing
(Contracting Agency)
Lt. Col. A. W. Dill (TWRD)
Lt. Col. M. S. McDowell (TWRU) (2)
TWACS (6)
V. Thomas

PIR-E1 (Lockheed)
N. N. Epstein
C. H. Finnie
H. F. Grover
H. R Miller
W. E. Moorman (5)
461 Program Office
698BK Program Office

PIR-E2 (Philco)
J. A. Bean
J. A. Isaacs
R. Morrison
S. M. Stanley

PIR-E3 (LFE)
D. F. Criley
K. B. Williams (5)

PIR-E8 (Mellonics)
F. Druding

PIR-E5 (Aerospace)
F. M. Adair
R. V. Bigelow
R. D. Brandsberg
L. H. Garcia
G. J. Hansen (3)
C. S. Hoff
L. J. Kreisberg
T. R Parkin
E. E. Retzlaff
H. M. Reynolds
D. Saadeh
R. G. Stephenson
V. White

PIR-E7 (STL)
A. J. Carlson (3)

PIR-E4 (GE-Sunnyvale)
J. Farrentine
N. Kirby

PIR-E4 (GE-Santa Clara)
D. Alexander

PIR-E4 (GE-Box 8555)
J. S. Brainard
R. J. Katucki
J. D. Selby

PIR-E4 (GE-3198 Chestnut)
J. F. Butler
H. D. Gilman

PIR-E4 (GE-Bethesda)
A. Pacchioli

PIR-E4 (GE-Box 8661)
J. D. Rogers

DISTRIBUTION LIST

Internal

NAME	ROOM	NAME	ROOM
AFCPL (5)	14059	Hillhouse, J.	24049
Allfree, D	22078	Holmes, M. A.	22082
Alperin, N.I.	24118A	Holzman, H. J.	22096B
Armstrong, E.	24089	Houghton, W. H.	22073
Bernards, R. M.	Sunnyvale	Hoyt, R. L.	14039
Biggar, D.	24090B	Imel, L. E.	14039
Bilek, R. W.	24124	Kastama, P. T.	24053
Black, H.	14039	Kayser, F. M.	25026
Brenton, L. R.	22070	Keddy, J. R.	25026
Burke, B. E.	22076	Key, C. D.	24123
Busch, R. E.	24065B	Keyes, R. A.	20073
Carter, J. S.	27032	Kinkead, R. L.	24071
Champaign, M E.	24127B	Kneemeyer, J. A.	24065A
Chiodini, C. M.	22078	Knight, R. D.	24110B
Cline, B. J.	24097	Kolbo, L. A.	24139
Cogley, J. L.	24135	Kostiner, M.	14056B
Conger, L.	22079	Kralian, R. P.	14039
Cooley, P. R.	24083	Kristensen, K.	Sunnyvale
Court, T. D.	22073	LaChapelle, F.	24061
Crum, D. W.	24093	Laughlin, J. L.	20073
Dant, G. B.	22073	LaVine, J.	20079
DeCuir, L. E.	22096A	Little, J. L.	20077
Derango, W. C.	24082B	Long, F.	24122
Dexter, G. W.	24128	Madrid, G. A.	22049
Disse, R. J.	24139	Mahon, G. A.	20076
Dobbs, G. H.	24094B	Marioni, J. D.	24076B
Dobrusky, W. B.	22125	Martin, W. P.	24089
Ellis, R. C.	24081	McKeown, J.	24121
Emigh, G. A.	14039	Michaelson, S. A.	14039
Ericksen, S. R.	24110A	Milanese, J. J.	24121
Felkins, J.	22070	Munson, J. B.	24048A
Foster, G. A.	14039	Myers, G. L.	14056A
Franks, M. A.	25030	Nelson, P. A.	24075
Frey, C R.	24049	Ng, J.	22049
Frieden, H. J.	24071	Ngou, L.	25030
Gardner, S. A.	22053	Padgett, L. A.	24085
Greenwald, I. D.	24058A	Patin, O. E.	Sunnyvale
Griffith, E. L.	27029	Polk, T. W.	24099
Haake, J. W.	24120	Pruett, B. R.	24073
Harris, E D.	24083	Raybin, M.	14039
Henley, D. E.	24058B	Reilly, D. F.	24085
Hill, C. L.	24057	Remstad, C. L.	27029

DISTRIBUTION LIST

internal

NAME	ROOM	NAME	ROOM
Rosenberg, E. J.	14050	Thompson, J. W.	22077
Russell, R. S.	14050	Thornton, R. L.	14050
Scholz, J W	14039	Totschek, R. A.	24090A
Scott, R. J.	24093	Vorhaus, A. H.	24076A
Seacat, C. M.	Sunnyvale	Wagner, I. T.	24081
Shapiro, R S.	25026	Warshawsky, S. B.	22082
Skelton, R. H.	24127A	West, G. D.	Sunnyvale
Solomon, J.	24053	West, G. P.	24094A
Speer, N. J.	20079	Wilson, G. D.	22101
Stone, E. S.	22116B	Winter, M. E.	24137
Tennant, T. C.	27024	Wise, R. C.	24051
Sweeney, M. J.	24057	Wong, J. P.	Sunnyvale
Testerman, W. D.	14039	Zubris, C. J.	24075

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
MILESTONE 11 OSAS-A MODIFIED FOR
AUGMENTATION (SOSAS).
Scientific rept., TM-1003/009/00,
15 March 1963, 52p.
(Contract AF 19(628)-1648, Space
Systems Division Program, for Space
Systems Division, AFSC)

Unclassified report

DESCRIPTORS: Satellite Networks.
Programming (Computers).

Reports that SOSAS (OSAS-A Modified for
Augmentation) allows the writing of

UNCLASSIFIED

---

programs for the 160-A computer in
symbolic notation with minimum
regard for ultimate storage locations
assigned a given program. States that
instructions using mnemonic symbols
can be written, these are easier to
remember and to interpret later than
are the octal machine code equivalents.
Reports that the primary difference
between SOSAS and OSAS-A is the handling
of a System Symbol Table by SOSAS.

UNCLASSIFIED

UNCLASSIFIED